7544 5757A

CONTROL DATA

Student Training Manual for

CYBER 175 FUNCTIONAL UNITS

VOLUME 1

An Individualized Course

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary Printing
1/11/82	
A	Manual Released
5/20/82	
	Printing, 5/20/83
	Printing, 11/15/83

Publication No.
75445737

QA EDUCATION, NTHESE
1009 Washington Ave. No.
Mpls., MN 55401

CONTENTS

CONTENTS (Contd)

Title Page

INTRODUCTION

Since they make all of your computations, it is important that you have a good understanding of the CYBER 170 Model 750/760 Central Processors Functional Units.

This course covers all nine of the functional units. While doing this you will review all of the CPU instructions, including Floating Point.

This course is individualized because CEs asked for courses that could be done in their area. Course time is at your discretion; the course uses very little computer time.

If this your first experience with a self-study or individualized course, read the procedure in appendix A. If not, turn to the module title page following this introduction and proceed as directed.

MODULE 1
POPULATION COUNT UNIT

Module 1 learning activities will acquaint you with the logical operation of the Population Count Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on to the CDC® PLATO ® terminal and take the module 1 test. Do not waste time on answers you do not know. By taking this test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column, put a check mark by each activity assigned to you by PLATO Learning Management (PLM) and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	1-A	Programmed Text: Population Count Instruction. This activity reviews the function of the Population Count instruction.	1-3
_____	_____	1-B	Text Reading: Population Count Primary Block Diagram. This activity traces the data bits and control flow through the 1.0 diagrams.	1-7

Assigned	Completed	Activity	Description	Page
_____	_____	1-C	Text Reading: Population Count Detailed Pak Diagrams. This activity traces the data bits and control flow through the 3.0 diagrams.	1-8
_____	_____	1-D	Exercise: Population Count Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the Population Count Functional Unit.	1-10

LEARNING ACTIVITY 1-A. PROGRAMMED TEXT: POPULATION COUNT INSTRUCTION

This activity reviews the function of the Population Count instruction. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 47 Population Count (Pop Count) instruction.

CPU FUNCTIONAL UNITS

Obtain the following reference material.

Title	Publication Number
6000 Code Card Book	60164500
CYBER 170 Computer Systems Code Book	60420010
CYBER 170 Model 176 Hardware Reference Manual	60456100
CYBER 170 Models 175, 740, 750, 760 Functional Units	GF60420300W

POPULATION COUNT INSTRUCTION

This instruction reads one operand from Xk, counts the number of one bits in the operand, and stores this 6-bit count in Xi. The count delivered to Xi is always a positive number or zeros are extended above the count. If the operand is all ones, a count of 60_{10} or 74_8 is delivered to Xi. If the operand is all zeros, a zero word is delivered to Xi.

An example of this is Xk = 0772 1234 0000 2601 0000.

There are 16_{10} one-bits so Xi = 0000 0000 0000 0000 0021.

CYBER 170 Model 750/760 Functional Units
Learning Activity 1-A

Directions: Work the following problem. Check your answer with those given at the end of the learning activity.

1. Initial conditions:

   (X3) = 0123 4567 7654 3210 5252
   Location 10000 = 47103 0000 0000 0000

   What is the contents of X1?

   30_{10} 36_8

   X1=

Refer to CYBER 170 (Model 720, 730, 750, and 760) Model 176 Hardware Reference Manual publication number 60456100, section 4 for more information.

The answer to learning activity 1-A is on the following page.

ANSWER FOR LEARNING ACTIVITY 1-A

1. X1 = 0000 0000 0000 0000 0036

You have completed learning activity 1-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 1-B. TEXT READING: POPULATION COUNT PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow through the Population Count Unit, primary block 1.0 diagram .

OBJECTIVE

- You will be able to follow the data and control flow through the Pop Count Unit, primary block 1.0 diagram.

Directions: Place microfiche GF60420300U Number 61 (10 of 10) in your microfiche viewer and select coordinate B3/B4.

The Population Count instruction requires two clock periods to complete execution. Data is received in the Population Count Unit from the CPU Xk Register during the same clock period that the 47 instruction is issued from the Current Instruction Word (CIW) Register. This 60 bits of data enters a First Stage Adder (1st Half Add), which partially sums the one bits. This partially reduced data is entered into the 27-bit Holding Register. This register can be loaded every clock period. The partial data is then sent to a Second Stage Adder (2nd Half Add), where it is summed to a 6-bit result. The output of the second stage is in complement form. When GO POP COUNT is received from the CPU Instruction Control it recomplements the 6-bit (0 through 5) result putting it in positive form and sends 54 zero-bits (6 through 59) to the CPU Xi Register. If GO POP COUNT is not received, the data in the Population Count Unit is never used.

You have completed learning activity 1-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 1-C. TEXT READING: POPULATION COUNT DETAILED PAK DIAGRAMS

This activity follows the data bits and control flow through the Population Count Unit detailed pak 3.0 diagram.

OBJECTIVES

- You will be able to follow the data and control flow through the Population Count Unit detailed pak 3.0 diagram.

Directions: Place microfiche GF60420300U Number 61 (10 of 10) in your microfiche viewer and select coordinate C1/C2.

CPU ADDER FUNCTIONS

The Population Count Unit adds a column of 60 bits. However, instead of adding all of those 60 bits at one time, the Pop Count Unit breaks the operand into three 6-bit sections and six 7-bit sections. During the first clock period of the operation, the First Stage Adder generates a partial sum of two partial sum bits and a partial carry bit for each of these nine sections. These 27 bits are loaded into the 27-bit Holding Register. During the second clock period these 27 bits go to the Second Stage Adder where the second adder combines the partial sum bits into a 6-bit result.

When GO POP COUNT is received from the CPU it gates the 6-bit output of the second stage adder through an AND NOT function. These six bits become bits 0 through 5 of Xi.

GO POP is also complemented and is fanned out to form the extended zero bits which form bits 6 through 59 of Xi.

In Quadrants A and C of the 3.0 diagram, the First Stage Adder is found on three KA modules (6B06 through 6B08). Each module receives 20 bits of the 60-bit operand from the CPU Operand Register Xk. Each module in turn is divided into three sections. Sections 1 and 2 are 7-bit sections and section 3 is a 6-bit section. Using the KA module at 6B06, section 1 has bits 0 through 6, section 2 has bits 7 through 13, and section 3 has bits 14 through 19.

In quadrant C, the KA module at 6B06 has section 1 and is also the First Stage Adder. The First Stage Adder network is made up of four adders. First bits 0 through 2 and bits 3 through 5 go to two 3-bit adders (X Adders). They are partially added here forming two SUM 0 and two CARRY 1 bits. Bit 6 is added to the two SUM 0 bits. This addition generates a single Sum 0 bit that goes to the Holding Register, and a CARRY 1 bit. These three CARRY 1 bits are added in the B Adder, which generates a SUM 1 and CARRY 2 bit that also go to the Holding Register.

On the second clock and still on the KA module, all three SUM 0 bits go to the A Adder, the three SUM 1 bits go the C Adder and the three CARRY 2 bits go to the D Adder. The SUMS and CARRY that are produced by these adds are the complement of the true result so are complemented and sent over to the KB module.

The KB module is a series of adders that pass on their SUM and CARRY bits to the next set of adders. After three stages of adders, a 6-bit total of all the one bits from Xk are in complement form. The results are gated to Xi through an AND NOT function by GO POP COUNT. GO POP COUNT passes through a fanout complementer, which generates all the zero bits 6 through 59 to Xi.

You have completed learning activity 1-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 1-D. EXERCISE: POPULATION COUNT TROUBLESHOOTING.

This activity allows you to develop approaches to troubleshooting the Population Count Functional Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W Number 61 (10 of 10) in your microfiche viewer and select coordinate C1/C2.

Since the whole Functional Unit consists of only four logic modules, there are no exercise problems. Whichever bits are failing swap out the KA modules first and then replace the KB module.

MODULE 2
LONG ADD UNIT

Module 2 learning activities aquaint you with the logical operation of the Long Add Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on to the PLATO terminal and take the module 2 test. Do not waste time on answers you do not know. By taking this test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column, put a check mark by each activity assigned to you by PLATO Learning Management (PLM) and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	2-A	Programmed Text: Long Add Instruction. This activity reviews the function of the Long Add instruction.	2-3
_____	_____	2-B	Text Reading: Long Add Primary Block Diagram. This activity traces the data bits and control flow through the 1.0 diagrams.	2-7

Assigned	Completed	Activity	Description	Page
______	______	2-C	Text Reading: Long Add Detailed Pak Diagrams. This activity traces the data bits and control flow through the 3.0 diagrams.	2-11
______	______	2-D	Exercise: Long Add Troubleshooting. This activity allows you to develop some basic methods of troubleshooting Long Add Add Functional Unit.	2-12

LEARNING ACTIVITY 2-A. PROGRAMMED TEXT: LONG ADD INSTRUCTION

This activity reviews the function of the Long Add instruction. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 36 and 37 Integer Sum and Integer Difference instructions.

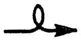
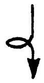
LONG ADD INSTRUCTIONS

Both of these instructions read their operands from two X registers (Xj and Xk). The result is a 60-bit integer sum or difference that is sent to operating Register Xi. All operands are signed integer. Overflow is not detected in either instruction.

In the Functional Unit prior to the addition, the unit complements both Xj and Xk operands. When subtracting, the unit complements just the Xj operand.

Following are two examples.

NOTE

The symbols  and  mean "complement to."

They are used often in Functional Unit instruction.

- 36123

  X2 = 0124 3570 6327 1120 5260

  X3 = 1000 5360 2342 3446 1120

  Complement the operands

  ```
  7653   4207   1450   6657   2517
  6777   2417   5435   4331   6657
  6652   6626   7106   3211   1376
  EAB                             1
  6652   6626   7106   3211   1377
  ```

  Complement to get back to true form below.

  X1 = 1125 1151 0671 4566 6400

- 37123

  X2 = 0025 7775 4351 5321 6240

  X3 = 0000 5374 6250 4302 1150

  Only complement Xj or X2

  ```
  7752   0002   3426   2456   1537
  0000   5374   6250   4302   1150
  7752   5377   1676   6760   2707
  ```

  Complement to get back to true form below.

  X1 = 0025 2400 6101 1017 5070

Directions: Work the following problem. Check your answers with those given on the following page.

1. Initial conditions:

```
(X4) = 0123  4567  7654  3210  5252
(X7) = 5252  2525  6161  1616  3333
(X6) = 4444  3333  2222  1111  0000
(X5) = 7676  2053  4760  2360  1472
```

Location 10000 = 36147 37265 0000 0000

What are the contents of X1 and X2?

```
7654 3210 0123 4567 2525
2525 5252 1616 6161 4444
2702 0462 1742 2750 7171
                       1
2702 0462 1742 2750 7172
5375 7315 6035 5027 0605
5375
```

a. X1 = 7315603550270605

```
3333 4444 5555 6666 7777
7676 2053 4760 2360 1472
2231 6520 2536 1247 1471
                       1
3231 6520 2536 1247 1472
```

b. X2 =

4546 1257 5241 65306305

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176 (Level B) Hardware Reference Manual, publication number 60456100, for more information.

ANSWERS FOR LEARNING ACTIVITY 2-A

```
1a.  X1 = 5375  7015  6135  5027  0605

     X4 = 0123 4567 7654 3210 5252 → 7654 3210 0123 4567 2525
     X7 = 5252 2525 6161 1616 3333 → 2525 5252 1616 6161 4444
                                     ------------------------
                                   ┌ 2402 0462 1742 2750 7171
                                   └→EAB                    1
                                     ------------------------
                                     2402 0462 1742 2750 7172

                                     Complement to get back to
                                     true form below.
                                                ↓
                                X1 = 5375 7315 6035 5027 0605

     X2 = 4546 1257 5241 6530 6305

     X6 = 4444 3333 2222 1111 0000 → 3333 4444 5555 6666 7777
     X5 = 7676 2053 4760 2360 1472   7676 2053 4760 2360 1472
                                     ------------------------
                                   ┌ 3231 6520 2536 1247 1471
                                   └→EAB                    1
                                     ------------------------
                                     3231 6520 2536 1247 1472

                                     Complement to get back to
                                     true form below.
                                                ↓
                                X2 = 4546 1257 5241 6530 6305
```

You have completed learning activity 2-A. You may review this material or continue with the next learning activity.

13 R

LEARNING ACTIVITY 2-B. TEXT READING: LONG ADD PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow through the Long Add Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Long Add Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 56 (5 of 10) in your microfiche viewer and select coordinate D3/D4.

The Long Add instruction requires two clock periods to complete execution. Data is received in the Long Add Unit from the CPU Xj and Xk Registers during the same clock that the 36 or 37 instruction is issued from the Current Instruction Word (CIW) Register. Note that both operands are complemented as they enter the functional unit. If a 37 instruction, the Xk operand is recomplemented. The first stage does a partial add (1st Half Add). On the second clock period this partial sum goes to the second stage and the addition is completed. When GO LONG ADD is received from the CPU Instruction Control it recomplements the 60 bits putting it in true form, and gating the data to Xi, the result register in the CPU.

Let's look at adder format and define some of the terms used (figure 2-1).

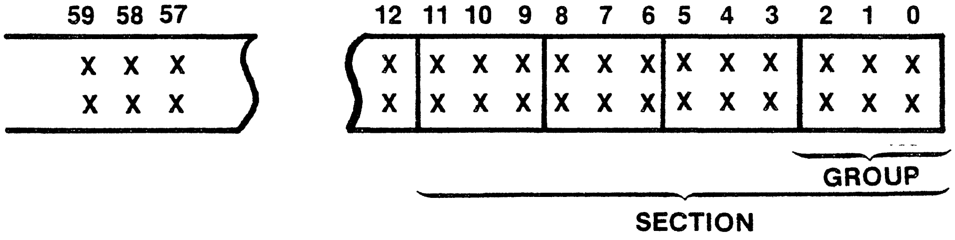


Figure 2-1. First Adder Format

1. GROUP = 3 bits

2. SECTION = 12 bits

Figure 2-1 shows that three bits make up a group and 12 bits make up a section.

FIRST STAGE

The 1.0 diagram shows the first stage forms a partial sum that consists of Bit Enables, Bit Borrow Generates, Group Borrow Generates, Section Borrow Generates, and Section Enables. Group Enables are also generated but are not sent to the second stage.

1. BIT ENABLE = $\begin{array}{c} 0 \\ \underline{1} \\ \text{ENABLE} \end{array}$ or $\begin{array}{c} 1 \\ \underline{0} \\ \text{ENABLE} \end{array}$

2. BIT BORROW GENERATE = $\begin{array}{c} 1 \\ \underline{1} \\ \text{BORROW (B)} \end{array}$

3. BIT SATISFY = $\begin{array}{c} 0 \\ \underline{0} \\ \text{SATISFY (S)} \end{array}$

4. GROUP BORROW GENERATE = A borrow out of a 3-bit group.

   Generates a bit out — Generates a bit out

   $\begin{array}{c} \text{xxx} \\ \text{xxx} \end{array}$ $\begin{array}{r} 110 \\ \underline{010} \\ 000 \end{array}$ or $\begin{array}{c} \text{xxx} \\ \text{xxx} \end{array}$ $\begin{array}{r} 101 \\ \underline{110} \\ 011 \end{array}$

5. GROUP ENABLE = All Bit Enables in the group are equal to a one.

6. SECTION BORROW GENERATE = A borrow out of a 12-bit section.

7. SECTION ENABLE = All group enables in the section are equal to a one.

SECOND STAGE

The second stage of the adder forms Group Borrow Inputs and Bit Borrow Inputs from the partial sum from the first stage. An EXCLUSIVE OR of the Bit Borrow Inputs and the Bit Enables form the final sum. This sum is in complement form. GO LONG ADD will gate the results through an AND NOT function to the CPU.

A Group Borrow Inputs equals a Borrow into the group, and a Bit Borrow Input equals a Borrow into a bit.

Table 2-1 is an example using only a 12-bit adder, to perform a 36 instruction. In this case, the Section Borrow Generate is the End-Around Borrow. In the case of the Long Add Unit, it is the Section Borrow from the highest order section.

TABLE 2-1. BIT, SECTION, AND GROUP GENERATION

4470 complements to 3307	011	011	000	111	
1532 complements to 6245	110	010	100	101	
	101	001	100	010	Bit Enables
	010	010	000	101	Bit Borrow Generates
	1	0	0	1	Group Borrow Generates
	1				Section Borrow Generates
	0				Section Enable
	0	0	1	1	Group Borrow Inputs
	100	100	001	111	Bit Borrow Inputs
	001	101	101	101	EXCLUSIVE OR of Bit Borrow Inputs and Bit Enables
	Complement to				
	110	010	010	010	
	6	2	2	2	

The example in table 2-1 shows not only all the different enables, generates, and inputs, it also shows the two times the bits are added. Taking the example we will place above each two bits what they constitute as too, an enable, borrow or satisfy. Then you will do the first half add, bring in the borrow inputs and then do the second half add.

EBE	SBE	ESS	BEB	
011	011	000	111	
110	010	100	101	
101	001	100	010	1st Half Add done in First Stage
100	100	001	111	Bit Borrow Inputs
001	101	101	101	2nd Half Add done in Second Stage
1	5	5	5	
6	2	2	2	

You have completed learning activity 2-B. You may review or continue with the next learning activity.

LEARNING ACTIVITY 2-C. TEXT READING: LONG ADD UNIT

This activity follows the data bits and control flow through the Long Add Unit detailed pak 3.0 diagrams.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Long Add Unit detailed 3.0 diagrams.

Directions: Place microfiche GF60420300W number 56 (5 of 10) in your microfiche viewer and select coordinate E1/E2.

LONG ADD UNIT

In Quadrant A and C of your 3.0 diagrams, the First Stage Adder is found on five KC modules (6B01 through 6B05). Each module receives 12 bits of the two 60-bit operands from the CPU Operand Registers Xj and Xk.

From the KC module at 6B01 for bits 0 through 11 or one section, both operands are complemented as they are received. If doing a 37 instruction, m0 would equal a one and would recomplement the Xk operand. The operands are then added in the First Stage Adder.

The first stage of the adder is divided into two parts. The first part develops the Bit Enables, Bit Borrow Generates, and Group Borrow Generates. These go to registers where they are held until the second clock period.

The second part of the First Stage Adder develops Section Enables and Section Borrow Generates.

This partial sum is sent to the five KD modules at 6C01 through 6C05, which also hold 12 bits per module. The Second Stage Adder is found on these modules. The result is in complement form. GO LONG ADD will gate the results through an AND NOT function before transmitting the results to Xi in the CPU.

You have completed learning activity 2-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 2-D. EXERCISE: LONG ADD TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the Long Add Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W Number 56 (5 of 10) in your microfiche viewer and select coordinate E1/E2.

This exercise gives you practice in troubleshooting problems in the Long Add Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the module(s) location and, if possible, the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction executed was a 37123.

```
Given:      X2 = 7777  7777  7777  7777  7777
            X3 = 0000  0000  0000  0000  0000
            X1 = 7777  7777  7777  7777  0000
```

6B01
6C01

2. Instruction executed was 36123.

```
   Given:      X2 = 0000  0000  0000  0000  0007
               X3 = 0000  0000  0000  0000  0000
               X1 = 0000  0000  0000  0000  0010
```

```
111 000
111 ΦΦΦ
-------
000 1 1 Φ
111
```

6B05

6C01

The answers to this learning activity are on the following page.

ANSWERS FOR LEARNING ACTIVITY 2-D

In the answers to these problems we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting we have included them.

1. The correct answer is:

   X1 = 7777 7777 7777 7777 7777

   Possible malfunction(s):

   a. 6B01 - Complement Gate Lower on KC module - pin 71 = 0

2. The correct answer is:

   X1 = 0000 0000 0000 0000 0007

   Possible malfunction(s):

   a. 6C01 - Section Borrow input failed on KD module
   TP64 = 1

   b. 6B05 - Section Borrow was not generated on KC module
   TP73 = 1

MODULE 3
INCREMENT UNIT

Module 3 learning activities acquaint you with the logical operation of the Increment Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on to the PLATO terminal and take the module 3 test. Do not waste time on answers you do not know. By taking this test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column, put a check mark by each activity assigned to you by PLATO Learning Management (PLM) and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	3-A	Programmed Text: Increment Instruction. This activity reviews the function of the Increment instruction.	3-3
_____	_____	3-B	Text Reading: Increment Primary Block Diagram. This activity traces the data bits and control flow through the 1.0 diagrams.	3-8
_____	_____	3-C	Text Reading: Increment Detailed Pak Diagrams. This activity traces the data bits and control flow through the 3.0 diagrams.	3-10
_____	_____	3-D	Exercise: Increment Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the Increment Functional Unit.	3-12

LEARNING ACTIVITY 3-A. PROGRAMMED TEXT: INCREMENT INSTRUCTION.

This activity reviews the function of the Increment instruction. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 50 through 57, 60 through 67, and 70 through 77 Increment instructions.

50 THROUGH 57 INSTRUCTIONS

The operands can come from any of the A, B, or X Registers. If it is a 30-bit instruction, 50 through 52, one of the operands will come from Big K. The result of the 50 through 57 instructions always goes to an Ai register. If the Result Register is A1 through A7, the lower 17 bits of the result are added to Reference Address Central (RAC) and sent to the Storage Address Stack (SAS). These two operations take place independently and in parallel with each other. If the result register equals A1 through A5, the result is a read from Central Memory to its corresponding X Register. If the result register is A6 or A7, it writes to Central Memory from its corresponding X Register.

The following are examples of these results.

- Location 10000 = 50120 00100 00000 00000.

  A2 = 000100

  ```
  You would have  A2 = 000100
                   K = 000100
                  A1 = 000200
  ```

  The 000200 would be added to RAC giving you an absolute address. This address would be referenced as a read and its contents would be stored in X1.

- Location 10000 = 56634 00000 00000 00000

  B3 = 010000 B4 = 005000

  ```
  You would have  B3 = 010000
                  B4 = 005000
                  A6 = 015000
  ```

  Again this result would be added to RAC. This address would be referenced as a write, and the contents of X6 would be written into Central Memory.

Directions: Work the following problem. Check your answers with those given at the end of the learning activity.

1. Initial conditions:

```
(B1) = 001234
(B2) = 076542
(B3) = 252525
(B4) = 011111
(B5) = 032012
(A0) = 666666
(A1) = 000001
(X4) = 00000 00000 00000 77777
(X7) = 25252 52525 25252 52525
Relative Location 00777 = 6666  6666  6666 6666 6666
Relative Location 10777 = 5555  5555  5555 5555 5555
Relative Location 43123 = 4444  4444  4444 4444 4444
Relative Location 10000 = 52740 01000 56012 56145
Relative Location 10001 = 00000 00000 00000 00000
Start execution of instructions starting at Relative
Address 10000.
```

What are the contents of the following registers and locations?

a. A7 = 10777

b. Location 010777 =

25252 52525 25252 52525

c. A0 =

```
 001234
 076542
-------
 077776
```

d. A1 =

```
 011111
 032012
-------
 043123
```

e. X1 = 44444444 4444 4444 4444

ANSWERS FOR LEARNING ACTIVITY 3-A

la,b. 52740 01000. You add Big K to X3 results to A7.

```
X4 = 00000 00000  00000  07777  Only use the lower 18
                                bits of X4

X4 = 007777
K  = 001000
A7 = 010777
```

Since the Result Register equals A7, you will write the contents of X7 to Relative Location 10777.

Location 10777 = 25252 52525 25252 52525

Register A7 = 010777

c. 56012 - You add Bl to B2 results to A0.

```
Bl = 001234
B2 = 076542
A0 = 077776
```

Since the Result Register equals A0, there will be no READS or WRITES to/from Central Memory.

Register A0 = 077776

d,e. 56145 - You add B4 to B5 result to Al.

```
B4 = 011111
B5 = 032012
Al = 043123
```

Since the Result Register equals Al, you will read the contents of Relative Address 043123 to Xl,
Xl = 4444 4444 4444 4444 4444.

Register Al = 043123

Register Xl = 4444 4444 4444 4444 4444

60 THROUGH 67, AND 70 THROUGH 77 INSTRUCTIONS

The operands can come from Big K or any register as they did in the 50 through 57 (5X) instructions. The difference in the Result Register for 60 through 67 (6X) instructions is the B Register and for the 70 through 77 (7X) instructions it is the X Registers. Also, there are no references made to Central Memory.

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, for more information.

You have completed learning activity 3-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 3-B. TEXT READING: INCREMENT PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow through the Increment Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Increment Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 61 (10 of 10) in your microfiche viewer and select coordinate E3/E4.

INPUT OPERAND SELECTION

The Increment instruction requires 23 clock periods to complete execution if it is a 5X instruction, or two clock periods if it is a 6X or 7X instruction. Data is received through five different inputs. In quadrant A, the first group of operand inputs from Aj, Bj, and Xj go to the Operand 1 Select. The operand selected is based on the value of the fm designator. The second group of operand inputs from Big K or Bk to the Operand 2 Select are in quadrant C. Again the operand selected here is based on the fm designator. Between the two Operand Selectors is the Instruction Translator, which receives the function and mode portion from Current Instruction Word (CIW) Register. The translation selects Aj, Bj, or Xj for operand 1 and Big K, $\overline{\text{Bk}}$ or Bk for Operand 2.

COMPUTATION

There are two adders and both are used simultaneously. The upper one in quadrant A and B has two inputs from your two operands. Like the other adders it does a first stage add generating Bit/Group Borrows and Bit/Group Enables, which are stored in the Partial Sum Holding Register. The second stage completes the addition in complement form. When GO INCREMENT is ANDed, with one of the three Select Registers signal, and the data will be gated and complemented to true form through the AND NOT function to either Ai, Bi, or Xi Result Registers in the CPU.

Every clock period the Increment Test Holding Register in quadrant B receives the output of the adder. The contents of this register are compared with Field Length (FL) from Central Memory in the CPU. If this value is equal to or greater than FL, the CM Direct Range Error Flag is set in the Program Status Designation (PSD) Register. This occurs only when a 50 through 57 instruction has caused a CM reference.

A second adder with three inputs is in quadrant D. The three inputs are the two operand inputs and RAC Central Memory. Like the other adder it stores the partial sum in a holding register. The addition is completed in the second stage and is in complement form. The result will be complemented and gated through an AND NOT function to the Storage Address Stack (SAS) when GO INCREMENT TO STORAGE is up. If the i designator in CIW has an octal value of zero, the GO INCREMENT TO STORAGE will not be present.

You have completed learning activity 3-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 3-C. TEXT READING: INCREMENT DETAILED PAK DIAGRAM

This activity follows the data bits and control flow through the Increment Unit detailed pak 3.0 diagrams.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Increment Unit detailed 3.0 diagram.

Directions: Place microfiche GF60420300W Number 61 (10 of 10) in your microfiche viewer and select coordinate F1/F2.

INPUT OPERAND SELECTION

In quadrants A and C of your 3.0 diagrams, the First Stage Adder is found on four KE modules (6K09 through 6K12). Each receives four bits of the 18-bit operand. There is one KF module (6K13) that receives bits 16 and 17.

The KE module at 6K09 receives bits 0 through 3. The top three inputs, Aj, Bj, and Xj are complemented and then go to an AND function. The Select Gate is determined by the m designator from the CIW in the Instruction Translator. The last two input lines on the module are Big K from CIW and Bk.

Big K is always complemented. With Bk, there is an option. If it is Select Bk, Bk will be complemented; if it is SEL Bk, KC will not be complemented. The Select Gates for the different inputs are determined by the designator m from CIW. The two selected operands go to both adders simultaneously.

COMPUTATION

At 6K09 the two operands enter the first stage, generating Group 0 Borrows and Enables, and bit 0 through 3 Borrows and Enables. These partial results go to the Partial Sum Holding Register. On the next clock, the second stage, the two KG modules (6K14 and 6K15), and a KH module at 6K16 complete the computation. Using the KG module at 6K14, function code bits f0 and f1, received from KF at 6K13, go to a translator that selects the correct result register A, B, or X. A X01 selects Ai, X10 selects Bi, and X11 selects Xi.

This translation is ANDed with GO INCREMENT to form three gates, XMIT to X, XMIT to A, and XMIT to B. The output of the second stage of the adder is gated through an AND NOT function by one of these three signals to the register in the CPU. The KH module at 6K16 also takes bit 17. This extends the sign of your result through bits 18 through 59 when the results go to the X Register.

The KI modules (6L14 and 6L15) in quadrant C make up the first stage of the three operand adder of the two operands and RAC. This partial sum is sent to the KJ module at 6L16 where the computation is done. When GO INCREMENT TO STORAGE the result is complemented through the AND NOT function. Bits 0 through 17 are sent to CMC as an absolute address.

INCREMENT TEST HOLDING REGISTER

The KK module at 6L12 in quadrant D holds a Relative Address and changes every clock period. This value is sent to the CPU and is checked against Field Length-Central Memory (FLS) to check for an address out of range. This can only happen when a 50 through 57 instruction has caused a CM reference.

You have completed learning activity 3-C. You may review or continue with the next learning activity.

LEARNING ACTIVITY 3-D. EXERCISE: INCREMENT TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the Increment Unit.

OBJECTIVE

- You will be able to analyze Increment Unit failures and come to a logical solution.

Directions: Place microfiche GF60420300W number 61 (10 of 10) in your microfiche viewer and select coordinate F1/F2.

Since the Increment Functional Unit consists of so few modules, there are no exercise problems.

MODULE 4
SHIFT UNIT

Module 4 learning activities acquaint you with the logical operation of the Shift Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on to the PLATO terminal and take the module 4 test. Do not waste time on answers you do not know. By taking the test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column below, put a check mark by each activity assigned to you by PLM and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	4-A	Programmed Text: Shift Unit Instruction. This activity reviews the function of the instruction.	4-3
_____	_____	4-B	Text Reading: Shift Primary Block Diagram. This activity traces the data bits and control flow through the 1.0 diagram.	4-9
_____	_____	4-C	Text Reading: Shift Detailed Pak Diagram. This activity traces the data bits and control flow through the 3.0 diagram.	4-12
_____	_____	4-D	Exercise: Shift Unit Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the Shift Functional Unit.	4-16

LEARNING ACTIVITY 4-A. PROGRAMMED TEXT: SHIFT UNIT INSTRUCTION

This activity reviews the function of the Shift instruction. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 20, 21, 22, 23, and 43 Shift instructions.

SHIFT INSTRUCTIONS

The operands can come from any of the X Registers. If it is a 20 or 21 instruction the operand is read from Xi and the Result Register is Xi. If a 22 or 23 the operand is read from Xk and the result goes to Xi. If a 43 instruction no operand is read, but the masking bits are sent to Xi.

20 Instruction

The operand is read from the Xi Register, shifts the 60-bit word left circularly by jk bit positions and writes the result back into the same Xi Register. The j and k designators are treated as a 6-bit positive integer operand.

The left circular shift means that bits are being moved to the highest order bit positions. The bits shifted off the upper end of the 60-bit word (bit 59) are reinserted in the lowest order bit (bit 0).

The resulting word has the same number of one and zero bits as in the original operand.

Following is an example of this operation.

- Location 10000 = 20112 00000 00000 00000

  X1 = 2323 6600 0000 0000 0111

  jk equals 12_8 or 10_{10}. Therefore,

  X1 = 7540 0000 0000 0022 2464

21 Instruction

This instruction operates like the 20 instruction except it is a right shift. The only other difference is that as the operand is shifted from highest to the lower order bits, the upper order bits are filled with copies of the original sign bit, bit 59. The bits shifted off from bit 0 are discarded.

Following are examples of this operation.

- Location 10000 = 21130 21210 00000 00000

  X1 = 2004 7655 0002 3400 0004

  jk equals 30_8 or 24_{10}. Therefore,

  X1 = 0000 0000 2004 7655 0002

  Since Bit 59 equals a zero, zeros are your sign extension.

  X2 = 6000 4420 2222 0000 5643

  jk equals 10_8 or 8_{10}. Therefore,

  X2 = 7774 0011 0404 4440 0013

  Since bit 59 equals a one, ones are your sign extension.

22 Instruction

This operand is read from the Xk Register and can be shifted either left or right and sends the results to Xi.

A left circular shift occurs when the value of Bj is positive (bit 17 clear or zero). This then is like a 20 instruction. A right shift occurs when the value of Bj is negative (bit 17 set or one). This then is like a 21 instruction.

If Bj bit 17 is set, only the lower 12 bits are used in determining the shift count. Bits 12 through 16 are not tested. Bits 0 through 11 are complemented and the resulting positive integer is the shift count. If the shift count is greater than 63_{10} or 77_8 the result stored in Xi consists of 60 copies of the original operand sign bit, bit 59.

If Bj is zero (000000 or 777777) the instruction will read the operand from Xk and send it unaltered to Xi.

If Bj is positive, only the lowest order six bits are used in determining the shift count. The remaining bits are ignored. For example, a shift count of 77 octal/63 decimal results in a left shift of three bit positions.

Following is an example of both positive and negative B registers.

```
●    Location 10000 = 22110  22223  00000  00000

     X0 = 2323  6600  0000  0000  0111
     B1 = 000012

     B1 = 12 octal or 10 decimal and it is positive so left
          shift 10 decimal

     X1 = 7540  0000  0000  0022  2464


     X3 = 1327  6000  0000  3333  2422
     B2 = 777771

     B2 complemented  = 6 octal or 6 decimal and bit 17 is
     set/negative, which means right shift.

     X2 = 0013  2760  0000  0033  3324
```

23 Instruction

This operand is read from the Xk Register and can be shifted either left or right and sends the results to Xi.

A right shift with a sign extension occurs when the value of Bj is positive. This then is like a 21 instruction. A left circular shift occurs when the value of Bj is negative. This then is like a 20 instruction.

If Bj bit 17 is clear and Bj bits 6 through 11 are different/set and the shift count is greater than 63_{10} or 77_8 places right, the result stored in Xi consists of 60 copies of the original operand sign bit, bit 59.

If Bj is zero (000000 or 777777) the instruction will read the operand from Xk and send it unaltered to Xi.

If Bj is negative, only the lowest order six bits are used in determining the shift count. The remaining bits are ignored. For example, a shift count of 77 octal/63 decimal results in a left shift of three bit positions.

Following is an example of both positive and negative B Registers.

- Location 10000 = 23110 23223 00000 00000

  X0 = 1327 6000 0000 3333 2422

  B1 = 000006

  B1 = 6 octal or 6 decimal and bit 17 is positive so right shift 6 decimal

  X1 = 0013 2760 0000 0033 3324

  X3 = 2323 6600 0000 0000 0111

  B2 = 777765

  B2 complemented = 12 octal or 10 decimal and Bit 17 is set/negative which means left shift.

  X2 = 7540 0000 0000 0022 2464

Refer to CYBER 170 (Model 720, 730, 750, and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, section four for more information.

43 Instruction

No operand is read from a register. The jk designators are treated as a 6-bit quantity which describes the size of the masking field. This field of one bits begins at the highest order bit of the 60-bit word, bit 59, and extends down toward bit 0. A completed masking word consists of one bits in the highest order bit positions and zeros in the remaining bit positions of the word. This masking word is then sent to Xi.

Following is an example of this instruction.

- Location 10000 = 43124 00000 00000 00000

  jk = 24 octal or 20 decimal, which means you will have 20 decimal ones.

  X1 = 7777 7760 0000 0000 0000.

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, section four for more information.

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

1. Initial conditions:

```
(B1) = 000030
(B2) = 777774
(X0) = 7435  1234  0000  2222  5353
(X1) = 0034  5252  2525  3333  0011
(X3) = 1234  5670  7654  3210  1111
(X6) = 6666  5555  4444  3333  2222

Location 10000 = 21014  20106  22213  23423
         10001 = 43614  00000  00000  00000
```

What are the contents of the following registers?

ok

a. X0 = 7777 7435 →

ok

b. X1 = 3452 →

ok

c. X2 = 7654 →

ok

d. X4 = 23456 →

ok

e. X6 = 7777 0 → 6

ANSWERS FOR LEARNING ACTIVITY 4-A

1a. 21014 RS X0 14_8 or 12_{10} places.

X0 = 7777 7435 1234 0000 2222

Have sevens in upper four digits because bit 59 was set so sign extend ones.

b. 20106 LS X1 6_8 or 6_{10} places

X1 = 3452 5225 2533 3300 1100

Left shift circular so six bits of zeros take up lower six bits.

c. 22213 LS nominal X3 to X2 by B1 or 30_8 or 24_{10} places.

X2 = 7654 3210 1111 1234 5670

d. 23423 RS nominal X3 to X4 by B2. B2 is negative so do an LS of 3 because 777774 complemented equals 000003.

X4 = 2345 6707 6543 2101 1111

e. 43614 - This says mask 14_8 or 12_{10} ones in X6.

X6 = 7777 0000 0000 0000 0000

You have completed learning activity 4-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 4-B. TEXT READING: SHIFT PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow through the Shift Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Shift Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 54, (3 of 10) in your microfiche viewer and select coordinate B3/B4.

INPUT OPERAND SELECTION

This shift instruction requires two clock periods to complete execution. In quadrant C of the 1.0 diagrams are the inputs from either Xi or Xk Registers that were selected in the CPU. The 60 bits of data can take one of two paths. If it is a right shift the data is gated by RIGHT SHIFT and enters the Operand Holding Register unchanged. On a left shift the data is gated by LEFT SHIFT and is left shifted circularly three bit positions before entering the Operand Holding Register. If doing a mask instruction, there are no gates, so the operand is discarded which gives all zeros in the Operand Holding Register.

After the Holding Register is Complement Control. This function will complement the 60 data bits if the selected operand was negative or it was a mask instruction.

SHIFT COUNT SELECTION

Bj Register input and jk from CIW are in quadrant A. Both are 6-bit inputs. For the 20, 21, or 43 instruction jk is used, and for 22 or 23 instructions the lower 6 bits of the Bj Register are used. If a 20 or 22 left shift instruction has been issued, the shift count bits are complemented in Complement Control.

INSTRUCTION TRANSLATION AND SHIFT COUNT GREATER THAN 63.

The Instruction Translator is in quadrant A. It uses f1, m0, and m1 from CIW and also bit 17 from Bj Register. These four bits are all that is needed to translate your shifts. From the translator, take the 22 or 23 instruction signal to Shift Count greater than 63

Test. If doing a right shift this circuit checks bits 6 through 11 and 17 of the Bj Register. If bits 6 through 11 were different from bit 17, your shift count is too great and you will not generate ENABLE SHIFT so there will be no output from the Rank 3 Shift Count Translator. No output causes none of the Rank 3 Shift Count Flags to set. When this occurs, the Shift Rank 3 outputs all zeros. When gated by GO SHIFT, these zeros are complemented by the AND NOT function, sending all ones to Xi. If the original operand was positive, it will be recomplemented before entering the Xi Register.

SHIFT COUNT TRANSLATORS AND FLAGS

Complement Control in quadrant A has Shift Count Bits selected into three groups. Bits 0 and 1 go to the Rank 1 Shift Count Translator and Flags. This translates to right shift 0, 1, 2, or 3-bit positions for shift Rank 1. Shift Count bits 2 and 3 go to quandrant B and the Rank 2 Shift Count Translator and Flags. This translates to right shift 0, 4, 8, or 12-bit positions for shift Rank 2 in quadrant D. Then Shift Count bits 4 and 5 go to Rank 3 Shift Count Translator through Enable Shift Gate to Rank 3 Shift Count Flags. These are translated to 0, 16, 32, and 48-bit positions for Shift Rank 3.

LEFT CIRCULATOR SHIFT SIMULATION

When executing a left shift instruction the shift unit actually executes a right circular shift. The Left Circular Shift Note 2 in quadrant D enables the three shift ranks to shift right circularly. The amount of shift is determined by the complement of the shift count. However, this introduces an error of three bit positions. An example of this could be a shift of 73 octal or 59 decimal. A left shift of 59 decimal must occur. This shift is the same as a right shift circular of one bit position, so bit 0 becomes bit 59. In reality, the complement of 73 is 04. This then would cause a right shift of four making bit 0 bit 56 instead. This error is corrected by the 3-bit Left Circular Shift Network, which left shifts the incoming operand three bits if doing a left shift.

SHIFT RANKS

Quadrant D shows the three shift ranks that shift the operand by the amount as determined by the Shift Count Flags. Each rank will shift right circular if left circular shift signal is present. If no signal is present, then it shifts right with sign extension. Since the operand is always positive the sign bit is always zero.

On a mask instruction the operand is blocked so that all zeros are loaded into the OPERAND HOLDING REGISTER. f1 will equal zero up through the Complementer and OR function. This one will now set COMPL. CONT. FLAG, which in turn sets COMP CONT. This complements all the zeros from the OPERAND HOLDING REGISTER to all ones in the RANK 1 SHIFT. It is then right shifted the amount of places which is determined by jK, placing zeros in the vacated upper bit positions. All outputs of the Shift Ranks are complemented while being transmitted to Xi in the CPU.

SHIFT UNIT OUTPUT

The output of the Shift Rank 3 is gated by GO SHIFT and is complemented through an AND-NOT function as it is transmitted to the Xi register. If the original operand was positive, this output is the complement of what the register wants, so the X Register circuitry in the CPU complements the Shift Units Output before storing the results in Xi. Negative operands have been complemented twice in the Shift Unit, so they are not complemented prior to storing in Xi.

You have completed learning activity 4-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 4-C. TEXT READING: SHIFT DETAILED PAK DIAGRAM

This activity follows the data bits and control flow through the Shift Unit detailed pak 3.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Shift Unit detailed 3.0 diagram.

Directions: Place microfiche GF60420300W number 54 (3 of 10) in your microfiche viewer and select coordinate C1/C2.

OPERAND SELECTION

In quadrant A and C are 10 KP modules 6A07 through 6A16. They all have a 12-bit input and 6-bit output. The labeling on the KP module at 6A16 shows bits 0 through 2 and bits 51 through 59 coming in and bits 54 through 59 leaving. Bits 51 through 53 came in because of the automatic three-position left shift at the beginning of the left shift. Bits 0 through 2 are not used on 6A07 in quadrant C which shows bits 57 through 59 and bits 0 through 8 coming in and bits 0 through 5 leaving. Bits 57 through 59 would be for the three-position left shift.

The KP module at 6A16 receives bits 51 through 59 and bits 0 through 2 from the Xi or Xk Registers. Assuming a right shift and a positive number, bit 54 would enter the module taking the top path through the Right Shift Gate to the Operand Holding Register.

If you assumed a positive number, bit 59 would not be set so the Complement Control Flag would not set Complement Control. Bit 54 would still be bit 54 in the Rank 1 Shift 0, 1, 2, or 3. If a 43 instruction, f1 becomes a zero and it will cause both the left and right shift gates to become zeros. This causes the operand to be discarded causing the Operand Holding Register to hold all zeros. The function code f1 is complemented and will set the Complement Control Flag. This will cause ones to be inputted to Rank 1 Shift 0, 1, 2 or 3. If jk equals 3, it causes the upper three bits of Rank 1 Shift bits 57 through 59 to be ones and the remaining 57 bits to be zeros.

SHIFT COUNT SELECTIONS

In quadrant A, the KO module at 6E10 is the Shift Count Translator. The amount of the shift was determined by the six-bit count from one of two sources. First is j and k from the CIW Register that is from CPU 3.1A and 3.1D. The instruction translation uses just f1, m0, and m1, also from the CIW Register on CPU 3.1A. If doing a 20, 21, or 43 instruction, jk will pass to Complement Control and be complemented if doing a 20 left shift instruction. These shift count bits go to Rank 2 shift 0, 4, 8, or 12 and Rank 3 shifts 0, 16, 32, 48 Translators (or Rank 1 Fanout). Rank 1 Fanout bits 0 and 1 go over to KP at 6A16 where they are translated. For Rank 1 Shift Count Flags are translated as shifts 0, 1, 2, or 3.

If doing a 22 or 23 instruction in quadrant A and the KO module at 6E10, the shift count is the lower six bits of Bj. The logic also checks bits 6 through 11 and bit 17. For bits 0 through 5 and the 22+23 gate, the shift count bits are sent to an OR function. From here it is the same as the 20 or 21 instruction.

SHIFT COUNT TRANSLATORS AND FLAGS

The shift count bits have gone to the Shift Count Translators on the KO module at 6E10 and the KP module at 6A16. The different Rank Shift Count Flags are on the KP modules. For example, if doing a shift of 12, this is on the KP at 6A11 Rank 2 Shift 12 Flag. Since Rank 1 has been covered, let's look at Ranks 2 and 3. In quadrant D are the KQ and KW modules at 6B09 through 6B16. These hold the Rank 2 Shift 0, 4, 8 and 12 and Rank 3 Shift 0, 16, 32, and 48 logic. Study notes 1 and 2 under the NOTE section on these modules.

SHIFT COUNT GREATER THAN 63 TEST

Prior to a 22 or 23 instruction, this network checks to see if shift count is greater than 63. This is only checked if shifting right. This logic is found on the KO module a 6E10. Bj bits 0 through 11 and bit 17 are received and the circuit looks for bit 17 being different from bits 6 through 11.

Assume a 23 instruction. Bit 6 is set and bit 17 is clear. Use Bj bit 17 (which equals zero), go through the complementor (now bit 17 equals one) up to the AND NOT function. Entering the AND NOT function at 12 o'clock is 23 instruction, which equals a one.

Bit 6 enters the AND-NOT function at 9 o'clock and would equal a one. With three ones in, Enable Shift equals zero. This blocks any Rank 3 translations so the Rank 3 Shift outputs all zeros. (For more information, read note 8 in the NOTE section.)

LEFT-CIRCULAR SHIFT SIGNAL AND FLAG

On the KO module the Left Circular Shift Flag goes to the KP module at 6A07. This signal is formed by the OR function. It has the following three inputs. The first, at 12 o'clock, is the 20 instruction. The second is the 22 instruction Left Shift Nominal, which needs a positive Bj bit 17. So Bj bit 17 is complemented and it makes the gate. The third, at 6 o'clock, is the 23 instruction Right Shift Nominal. To left shift, it needs Bj bit 17 set. If Bj bit 17 is set, the gate is made for a left shift.

SHIFT UNIT OUTPUT

The Rank 3 Shift KQ and KW modules output the shifted operand through an AND NOT function when GO SHIFT is received from the CPU. The operand is in complement form if the original operand was positive but will be recomplemented by X Register Control before being stored in Xi.

Consider the following example.

Initial conditions:

Assume 21130, X1 will be right shifted 30_8 or 24_{10} places. Also assume X1 equals 2004 7655 0002 3400 0004. At the completion, X1 should equal 0000 0000 2004 7655 0002. Take bit 25 and shift this to the right. The shift network will have shifts of 8 and 16 up.

Solution:

Bit 25 from Xi will enter 6A11 as bit 25 through the Rank 1 Shift of 0 to bit 25. Bit 25 goes to KQ at 6B14 Rank 2 Shift 8 to bit 17. Bit 17 goes to KQ at 6B10 Rank 3 Shift 16 to bit 1. This bit is then output to Xi.

You have completed learning activity 4-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 4-D. EXERCISE: SHIFT UNIT TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the Shift Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W number 54, (3 of 10) in your microfiche viewer and select coordinate C1/C2.

This exercise gives you practice in troubleshooting problems in the Shift Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the module(s) location and if possible the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction executed was a 20106.

```
   Given:        X1 initial = 0777 7777 7777 7777 7777
                 X1 result  = 0777 7777 7777 7777 7777
```

2. Instruction executed was a 20106.

```
Given:     X1 initial = 0777 7777 7777 7777 7777
           X1 result =  0000 0000 0000 0000 0007
```

3. Instruction executed was a 23123.

```
Given:     X3 = 4000 0000 0000 0000 0000
           B2 = 000014
           X1 = 7777 3700 0000 0000 0000
```

4. Instruction executed was a 22123.

```
Given:     X3 = 0777 7777 7777 7777 7777
           B2 = 000003
           X1 = 7376 7757 7376 7757 7370
```

5. Instruction executed was a 22123.

```
Given:     X3 = 0777 7777 7777 7777 7777
           B2 = 777771
           X1 = 0007 7777 7777 7777 7403
```

The answers to this learning activity are on the following page.

ANSWERS FOR LEARNING ACTIVITY 4-D

In the answers to these problems we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting we have included them.

1. The correct answer is:

   X1 = 7777 7777 7777 7777 7707

   Possible malfunction(s):

   a. 6E10 - Select jk network KO module

2. The correct answer is:

   X1 = 7777 7777 7777 7777 7707

   Possible malfunction(s):

   a. 7A07 - Left circular signal KP module

3. The correct answer is:

   X1 = 7777 4000 0000 0000 0000

   Possible malfunction(s):

   a. 6A16 - Complement gate KP module

4. The correct answer is:

X1 = 7777 7777 7777 7777 7770

Possible malfunction(s):

a. 6B09 . Gate output KQ module

5. The correct answer is:

X1 = 0007 7777 7777 7777 7777

Possible malfunction(s):

a. 6A08 - Complement gate KP module

b. 6A08 - Shift 2 term KP module

MODULE 5
BOOLEAN UNIT

Module 5 learning activities acquaint you with the logical operation of the nonfloating point portion of the Boolean Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on to the PLATO terminal and take the module 5 test. Do not waste time on answers you do not know. By taking this test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column below, put a check mark by each activity assigned to you by PLM and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	5-A	Programmed Text: Boolean Instruction. This activity reviews the function of the Boolean instruction.	5-3
_____	_____	5-B	Text Reading: Boolean Unit Primary Block Diagram. This activity traces the data and control flow through the nonfloating point portion of the 1.0 diagram.	5-7

Assigned	Completed	Activity	Description	Page
_____	_____	5-C	Text Reading: Boolean Detailed Pak Diagram. This activity traces the data and control flow through the nonfloating point portion of the 3.0 diagram.	5-11
_____	_____	5-D	Exercise: Boolean Unit Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the nonfloating portion of the Boolean Functional Unit.	5-14

LEARNING ACTIVITY 5-A. PROGRAMMED TEXT: BOOLEAN INSTRUCTION

This activity reviews the function of the Boolean instruction. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 10 through 17 Boolean instructions.

10 AND 14 INSTRUCTIONS

These two instructions transfer a 60-bit word from an Xj Register if it is a 10 instruction to an Xi Register, or from an Xk Register if it is a 14 instruction to an Xi Register. There is no logical function performed on the data. The difference between the two is that the 14 instruction complements the 60-bit word from Xk before sending the data to Xi.

Following is an example of this instruction.

- Location 10000 = 10130 14203 00000 00000

  X3 = 2525 5252 7777 4444 1111

  transfer X3 to X1. Therefore,

  X1 = 2525 5252 7777 4444 1111

  The 14 instruction wants X3 transferred to X2 and complemented. Therefore, X2 = 5252 2525 0000 3333 6666

11 THROUGH 13 AND 15 THROUGH 17 INSTRUCTIONS

These instructions require logical operations. They get their operands from the Xj and Xk Registers and send the results to the Xi Register. There are three different functions in this group. The 11 and 15 instructions are logical product or AND functions. The only difference is that the 15 instruction complements operand Xk before doing the logical product. A look at four bits of the operands shows the following results.

11 instruction	Xj = 0101	15 instruction	Xj = 0101
	Xk = 1100		$\overline{\text{Xk}}$ = 0011
	Xi = 0100		Xi = 0001

The second function is logical sum. These are the 12 and 16 instructions or INCLUSIVE OR functions. The only difference is that the 16 instruction complements operand Xk before doing the logical sum. A look at four bits of an operand shows the following results.

```
12 instruction    Xj = 0101      16 instruction    Xj = 0101
                  Xk = 1100                        X̄k = 0011
                  Xi = 1101                        Xi = 0111
```

The third function is logical difference. These are the 13 and 17 instructions or EXCLUSIVE OR functions. Here, as before, the only difference is the 17 instruction complements Xk. A look at four bits of this operand shows the following results.

```
13 instruction    Xj = 0101      17 instruction    Xj = 0101
                  Xk = 1100                        X̄k = 0011
                  Xi = 1001                        Xi = 0110
```

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

1. Initial conditions:

```
(X1) = 7777 7000 0123 4567 1010
(X2) = 0123 4567 0077 7700 1100
(X5) = 0000 7777 0123 4567 1010
(X6) = 0123 4567 7777 0000 1100
```

Location 10000 = 10010 11320 16456 13756

What are the contents of the following registers?

a. X0 = 77777 000 0123 4567 1010

X2=01234567 0077 77001100

b. X3 = 01234 0060 0023 45001000

c. X4 = ______________

d. X7 = ______________

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, section 4 for more information.

C) 0000 7777 0123 4567 1010
7654 3210 0000 7777 6677
7654 7777 0123 7777 7677

D) 0123 3210 7654 4567 0110

ANSWERS FOR LEARNING ACTIVITY 5-A

1a. 10010 transmits X1 to X0

```
X1 = 7777 7000 0123 4567 1010 so X0 will be the same value
```

b. 11320 Logical product of X2 and X0 to X3

```
X0 = 7777 7000 0123 4567 1010
X2 = 0123 4567 0077 7700 1100
X3 = 0123 4000 0023 4500 1000
```

c. 16456 Logical sum of X5 and the complement of X6 to X4

```
X5 = 0000 7777 0123 4567 1010 → 0000 7777 0123 4567 1010
X6 = 0123 4567 7777 0000 1100 → 7654 3210 0000 7777 6677
X4 =                            7654 7777 0123 7777 7677
```

d. 13756 Logical difference of X5 and X6 to X7

```
X5 = 0000 7777 0123 4567 1010
X6 = 0123 4567 7777 0000 1100
X7 = 0123 3210 7654 4567 0110
```

This completes learning activity 5-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 5-B. TEXT READING: BOOLEAN UNIT PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow through the nonfloating point portion of the Boolean Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the nonfloating portion of the Boolean Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate B3/B4.

The Boolean instruction requires two clock periods to complete execution. The Boolean Unit executes instructions that require bit by bit operation. This includes both logical and transmissive operations. Instructions 11, 12, 13, 15, 16, and 17 require logical operations; 10 and 14 are transmissive operations.

INPUT REGISTERS

There are three input registers in quadrant A and C. The Xj Register receives all 60 bits of the Xj operand from the CPU. The Xk operand (also from the CPU) goes to two different registers. Bits 48 through 59 go to the Xk Exponent Register and bits 0 through 47 go to the Xk Coefficient Register. These are clear/enter type registers. New data is entered every clock period regardless of the instruction being executed. These operands can then be used in the Boolean Unit the following clock period.

CONTROL

The Instruction Control Translator is in quadrant A. It receives f0, m0, ml, and m2 from the CIW Register. During each clock period the translator determines the type of logical operation and selects the proper paths for that operation. GO BOOLEAN from the CPU gates the results through an AND NOT function to Xi.

The following control signals are generated by the Instructor Control Translator for logical and transmit instructions.

Signal	Instruction
GATE Xk UPPER	12+13+14+16+17
COMP Xk	14+15+16+17
GATE Xj	10+12+13+16+17
GATE LP	11+12+15+16
GATE Xk LOWER	12+13+14+16+17+26+27

The following is an explanation of what the preceding control signals do.

GATE Xk UPPER — Gates bits 48 through 59 of the Xk operand into the Equivalent Circuit for 12, 13, 14, 16, and 17 instructions.

COMP Xk — Complements Xk for 14, 15, 16, and 17 instructions.

GATE Xj — Gates bits 0 through 59 of the Xj operand into the Equivalence Circuit for 10, 12, 13, 16, and 17 instructions.

GATE LP — Gates the logical product of the Xj and Xk operands for 11, 12, 15, and 17 instructions.

GATE Xk LOWER — Gates bits 0 through 47 of the Xk operand into the Equivalence Circuit for 12, 13, 14, 16, 17, 26, and 27 instructions.

METHOD

If GO BOOLEAN sets, a Boolean instruction was issued during the previous clock period. The data in the Xk Exponent and Coefficient Registers and the Xj Registers must be the operand as described by the j and k designators in that instruction. Data from these registers are merged in a static network and transmitted to the Xi Register. The type of logical operation and the selection of the data paths in this static network is set by Instruction Control Translator.

Logical Product (AND function) 11 AND 15 Instruction

The logical product is formed from the Xj Register and Xk Register (or its complement) and the result is transmitted to the Xi Register.

This operation has Xj register bits going to an AND function in quadrant D. The Xk exponent and Coefficient Register bits can be complemented as in the 15 instruction and go to the same AND function in quadrant D. The GATE LP control enters and completes the AND function. This result is complemented, but is recomplemented in an AND NOT function as it is gated by GO BOOLEAN and transmitted to Xi.

The following are examples.

$$\begin{array}{lll} \text{Xj} = 0101 & \qquad & \text{Xj} = 0101 \\ \underline{\text{Xk} = 1100} & & \underline{\overline{\text{Xk}} = 0011} \\ \text{Xi} = 0100 & & \text{Xi} = 0001 \end{array}$$

Logical Sum (INCLUSIVE OR Function) 12 and 16 Instruction

The Logical Sum is formed from the Xj Register and the Xk Register (or its complement) and this result is transmitted to the Xi Register.

This operation is done in two steps. First, Xk Exponent and Coefficient Registers are gated by Xk Upper and Xk Lower and goes to the Equivalence Circuit in quadrant D. The Xj Register is gated by GATE Xj also to the Equivalence Circuit. Second, the Equivalence Circuit output is ANDed with the complement of the Xk Register as gated by GATE LP. This output is also complemented as it is gated and transmitted to the Xi Register by GO BOOLEAN through an AND NOT function.

The following are examples.

$$\begin{array}{lll} \text{Xj} = 0101 & \qquad & \text{Xj} = 0101 \\ \underline{\text{Xk} = 1100} & & \underline{\overline{\text{Xk}} = 0011} \\ \text{Xi} = 1101 & & \text{Xi} = 0111 \end{array}$$

Logical Difference (EXCLUSIVE OR Function) 13 and 17 Instruction

The Logical Difference is formed from the Xj Register and the Xk Register (or its complement) and this result is transmitted to the Xi Register.

This operation has the Xk Exponent and Coefficient Registers gated by both Xk Upper and Xk Lower and the Xj Register is gated by GATE Xj going to the Equivalence Circuit. This output is complemented and transmitted to the Xi Register by GO BOOLEAN through an AND-NOT function.

The following are examples.

$$\begin{array}{l} Xj = 0101 \\ \underline{Xk = 1100} \\ Xi = 1001 \end{array} \qquad \begin{array}{l} Xj = 0101 \\ \underline{\overline{Xk} = 0011} \\ Xi = 0110 \end{array}$$

Transmit Xj or Xk

The transmit operation is executed by sending either Xj Register as gated by GATE Xj or Xk Exponent and Coefficient Registers complemented and gated by GATE Xk Upper and GATE Xk Lower to the Equivalence Circuit. The other operand will be zero. This result is complemented and transmitted to the Xi Register by GO BOOLEAN through an AND NOT function.

You have completed learning activity 5-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 5-C. TEXT READING: BOOLEAN DETAILED PAK DIAGRAM

This activity follows the data bits and control flow through the nonfloating point portion of the Boolean Unit detailed pak 3.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Boolean Unit detailed 3.0 diagrams.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate C1/C2.

INPUT REGISTERS

Five KK modules (6C07 through 6C11) hold the Xj Input Register in quadrant C. Each receives 12 bits from the Xk Register in the CPU. The Xk Input Register uses only four of these KK modules (6C07 through 6C10). Xk bits 48 through 59 are received on a KM module at 6C15 in quadrant A. The remaining portion of these modules are used for the 26 and 27 instructions. This data is loaded in these registers every clock period no matter what instruction is being executed. On the next clock period the logical functions are executed.

CONTROL

The instruction translation is done on the KM module at 6C15 in quadrant A. It receives f0, m0, m1, and m2 every clock period from the CIW. These translations determine the logical function and the data paths required by the instruction. Refer to the chart on 6C15, for which gates are used for the different instructions. The GO BOOLEAN is received on the second clock period from the CPU and is all that is necessary to transmit the results to Xi.

METHOD

The logical function is executed on three KL modules (6C12 through 6C14) in quadrant D and the KN module at 6C16 in quadrant B. The KL modules receive 16 bits each from the Xj or Xk Input Registers. The KN module receives bits 48 through 59. This module is also used on

26 and 27 instructions. The following text only brings up the necessary gates. It shows data flow on the KL and KN modules and for most of this example uses the KL module 6C12.

Logical Product (AND Function) 11 and 15 Instruction

The instruction translation chart on KM at 6C15 shows that the 11 instruction brings up GATE LP only. Xk bits 0 through 15 go through Complement Control, but since it is not a 15 instruction, the data is not complemented.

The Xk bits drop down and enter the AND NOT function. Xj bits 0 through 15 come into the AND NOT function and with GATE LP complete the AND NOT function. This NOT result enters an AND function. The output of the Equivalence Circuit is all ones because the GATE Xk and Xj gates are not enabled on either an 11 or 15 instruction. This sends the results to another AND NOT function that complements the result when GO BOOLEAN comes up. This puts the result in true form and transmits the results to the Xi Register.

Logical Sum (INCLUSIVE OR Function) 12 and 16 Instruction

The translation chart on the KM module shows that on a 12 or 16 instruction GATE Xk Upper, GATE Xk Lower, GATE Xj, and GATE LP are used. This function is executed in two steps. First, Xk bits 0 through 15 go through Complement Control, but since it is not a 16 instruction, the data will not be complemented. They then drop down and enter an AND NOT function. Xj bits 0 through 15 also enter this AND NOT function. GATE LP and the AND NOT function gate this partial result to the AND function just as in logical product. In the second step, the Xk bits 0 through 15 are gated to the Equivalence Circuit by GATE Xk Lower. Xj bits 0 through 15 are gated to the Equivalence Circuit by GATE Xj. This circuit output is ANDed with the first result and is complemented and gated to Xi by GO BOOLEAN through the AND NOT function.

If, for example, Xj equals 0101 and Xk equals 1100, the result in Xi should equal 1101. The operations take place in the following manner.

First, you would AND: $\begin{array}{l} \text{Xj} = 0101 \\ \underline{\text{Xk} = 1100} \\ \text{LP} = 0100 \end{array}$

This 0100 would be complemented to equal 1011.

Second, you do an EQUIVALENCE: $\begin{array}{l} \text{Xj} = 0101 \\ \underline{\text{Xk} = 1100} \\ \text{EQ} = 0110 \end{array}$

These two results are then ANDed:

```
LP = 1011
EQ = 0110
Xi = 0010
```

This result is complemented as it is gated and transmitted to Xi through the AND NOT function. This results in Xi is equal to 1101.

Logical Difference (INCLUSIVE OR Function) 13 and 17 Instruction

The translation charts show that on a 13 or 17 instruction you use GATE Xk Upper, GATE Xk Lower, and GATE Xj. As in the 12 instruction, Xk bits 0 through 15 and Xj bits 0 through 15 are gated to the Equivalence Circuit. Since GATE LP is not enabled, all ones will be ANDed with this result. This in turn is complemented as it is gated and transmitted to Xi by GO BOOLEAN through the AND NOT function.

Transmit Xj or Xk

The execution of either of these is the same with the exception of complementing Xk. Either operand is sent to the Equivalence Circuit with the other operand equal to a zero because it is not enabled. The output is ANDed with all ones because GATE LP is not enabled. This is gated and complemented as in the previous instructions.

You have completed learning activity 5-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 5-D. EXERCISE: BOOLEAN UNIT TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the nonfloating point portion of the Boolean Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate C1/C2.

This exercise gives you practice in troubleshooting problems in the nonfloating point portion of the Boolean Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the module(s) location and if possible the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction executed was a 11123.

```
   Given:    X2 = 7777 7777 7777 7777 7777
             X3 = 7777 7777 7777 7777 7777
             X1 = 7777 7777 7777 7760 0000
```

6C12 4K27

2. Instruction executed was a 15123.

```
Given:    X2 = 7777 7777 7777 7777 7777
          X3 = 7777 7777 7777 7777 7777
          X1 = 0000 0000 0000 0017 7777
```

6C12 4K77

The answers to this learning activity are on the following page.

ANSWERS FOR LEARNING ACTIVITY 5-D

In the answers to these problems we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting we have included them.

1. The correct answer is:

   X1 = 7777 7777 7777 7777 7777

   Possible malfunction(s):

   a. 6C12 = Complement gate on KL module - TP21 and TP22 = 0

   b. 6C12 = Gate LP term on KL module - TP55 and TP56 = 1

2. The correct answer is:

   X1 = 0000 0000 0000 0000 0000

   Possible malfunction(s)

   a. 6C12 - Complement gate on KL module - TP21 and TP22 = 1

   b. 6C12 - Gate Xj term on KL module - TP45 and TP46 = 0

   c. 6C12 - Gate output on KL module - TP11 and TP12 = 1

MODULE 6
BOOLEAN UNIT

Module 6 learning activities acquaint you with the logical operation of the floating point portion of the Boolean Functional Unit and review some floating point terminology and the function of the instruction.

PRETEST

Before you begin this module sign on to the PLATO terminal and take the module 6 test. Do not waste time on answers you do not know. By taking this test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column below, put a check mark by each activity assigned to you by PLM and then proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	6-A	Programmed Text: Pack and Unpack Instructions. This activity reviews some floating point terminology and the function of the Pack and Unpack instruction.	6-3
_____	_____	6-B	Text Reading: Boolean Primary Block Diagram. This activity traces the data bits and control flow used by the 26 and 27 instruction through the 1.0 diagram.	6-25
_____	_____	6-C	Text Reading: Boolean Detailed Pak Diagram. This activity traces the data bits and control flow used by the 26 and 27 instruction through the 3.0 diagram.	6-28

Assigned	Completed	Activity	Description	Page
_____	_____	6-D	Exercise: Boolean Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the Pack and Unpack portions of the Boolean Functional Unit.	6-31

LEARNING ACTIVITY 6-A. PROGRAMMED TEXT: PACK AND UNPACK INSTRUCTIONS

This activity reviews floating point terminology and functions of the Pack and Unpack instructions. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 26 and 27 Boolean Instructions.

FLOATING POINT

The following is a review of floating point terminology, format, exponent sign bits, and the relationship between actual exponent values and the biased exponent values.

Terminology

Coefficient — 49 bits of a floating point operand (modulus $2^{49}-1$), which represents the number K in the floating point notation K. 2^0. The radix point is assumed to be to the right of the 2^0 bit.

Exponent — 11 bits of a floating operand (modulus $2^{11}-1$), which represents the number n in the floating point notation (K.2^n) into which the base number 2 is raised.

Pack — Identifies the process of forming a floating point operand from registers separately containing a coefficient and exponent.

Unpack — Identifies the process of forming a separate coefficient and exponent from a floating point operand.

Round — The process of increasing a coefficient value by a fractional amount (usually 2^{-1}) so that any later arithmetic combination resulting in a fraction, equal to or greater than 2^{-1}, will force a carry into bit 2^0 of the result. For limited accuracy then the whole number is most correct and the fractional results may be discarded.

Equalize — The process of right shifting the coefficient of the smaller exponent a number of places equal to the difference between the larger exponent and the smaller exponent to make the smaller exponent equal the larger exponent. This process is necessary prior to adding or subtracting floating point operands.

Single-Precision — Identifies the extraction of the most significant 48 bits (2^0 through 2^{47}) of a 96-bit coefficient result.

Double-Precision — The extraction of the least significant 48 bits (2^{-1} through 2^{-48}) of a 96-bit coefficient result.

End Case — A special hardware consideration that either flags or corrects an erroneous result operand.

Format

Figure 6-1 shows the actual layout of a floating point number.

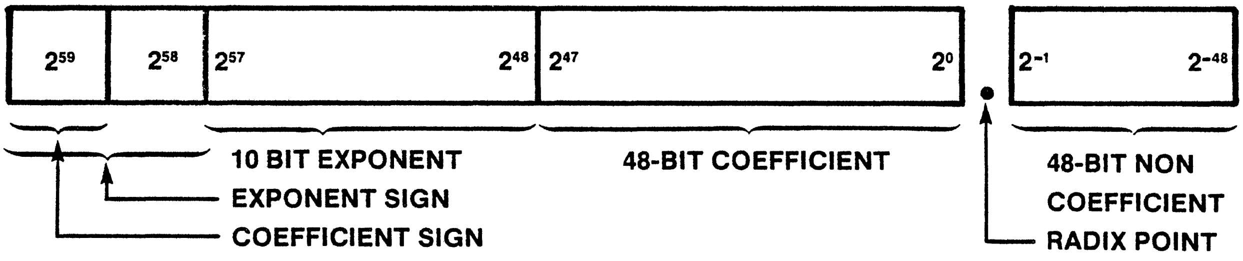


Figure 6-1. Floating Point Format

Bit group 2^{-1} through 2^{-48} is the noncoefficient or the double precision result. This area is used to generate an End-Around Carry into a single precision answer. The radix or decimal point is to the right of the Least Significant Bit (LSB).

The next 48 bits (2^0 through 2^{47}) are the coefficient or the number.

The next 10-bit group (2^{48} through 2^{57}) is the exponent or the power of the number. Bit 2^{59} is the sign of the coefficient and the combination of 2^{59} and 2^{58} is the sign of the exponent. The four combinations that must be satisfied with the sign bits are shown in table 6-1.

TABLE 6-1. EXPONENT SIGN BITS

Coefficient	Exponent	Bit 59	Bit 58
Positive	Positive	0	1
Positive	Negative	0	0
Negative	Negative	1	1
Negative	Positive	1	0

When the coefficient is positive, bit 59 equals zero and one if negative. If the exponent is a positive value, bits 59 and 58 are not equal, but they will be equal if the exponent is negative.

Table 6-2 illustrates how the exponent looks in all four cases over the full range of exponent values.

Since the size of the exponent is 10 bits, the value can range from 0000 through 1777. A positive coefficient and positive exponent in table 6-2 shows actual exponent value goes from +0000 to +1777. Bit 59 is not equal to bit 58, so the exponent is biased by setting 2^{10} of the exponent or 2^{58} of the 60-bit word. This gives a range of 2000 through 3777 or the biased exponent value. Throughout this range of numbers bit 2^{59} is not equal to bit 2^{58}. This denotes a positive exponent.

For a positive coefficient and a negative exponent bit 2^{59} must be equal to zero again. Bit 2^{59} must also equal bit 2^{58}. If you complement the actual value of -0000 to -1777, it denotes a negative number in a register. Since Bit 2^{59} equals zero for positive number, and bit 2^{59} is equal to bit 2^{58} for a negative exponent, the biased exponent value will be 1777 through 0000 for an actual exponent value of -0000 through -1777. The biased exponent value then for a negative exponent of -0000 and a positive coefficient is 1777.

TABLE 6-2. FLOATING POINT NUMBERS

Coef Sign	Exp Sign	Actual Exp Value	Biased Exp Value	Special Case
Negative	Negative	-1777	7777	-Underflow
		-1776	7776	
		.	.	
		.	.	
		.	.	
		.	.	
		0001	6001	
		0000	6000	Indefinite (-IND)
Negative	Positive	+0000	5777	
		+0001	5776	
		.	.	
		.	.	
		.	.	
		.	.	
		+1776	4001	
		+1777	4000	Infinity (-00)
Positive	Positive	+1777	3777	+Infinity (+00)
		+1776	3776	
		.	.	
		.	.	
		.	.	
		.	.	
		+0001	2001	
		+0000	2000	
Positive	Negative	-0000	1777	+Indefinite (+Ind)
		-0001	1776	
		.	.	
		.	.	
		.	.	
		.	.	
		1776	0001	
		1777	0000	+Underflow

For a negative coefficient and a negative exponent, bit 2^{59} must equal a one and bit 2^{59} must be equal to bit 2^{58}.

For a positive coefficient and a negative exponent, you saw that the biased exponent value was 1777 for an actual exponent value of -0000. Since this is a negative number, 1777 is complemented to 6000. For an actual exponent value of -0000, the biased exponent value would equal 6000. Bit 2^{59} equals 1 for a negative number and bit 2^{59} equals bit 2^{58} for a negative exponent.

For a negative coefficient and a positive exponent bit 2^{59} must equal a one and bit 2^{59} must not equal bit 2^{58}.

For a positive coefficient and positive exponent, the bias exponent value was 2000 for an actual exponent value of +0000. This is complemented because you have a negative coefficient, so 2000 becomes 5777. Bit 2^{59} equals 1 for a negative coefficient and Bit 2^{59} is not equal to bit 2^{58} for a positive exponent.

To find the actual exponent value of any biased exponent, first make it a positive coefficient. If this value ranges from 2000 through 3777, subtract 2000; if the value ranges from 0000 through 1777, subtract this value from 1777.

TABLE 6-3. EXAMPLES OF THE FOUR POSSIBLE CONDITIONS

Biased Value =	Actual Value	Result
1. 2007 =	2007-2000 or +0007	Positive Coefficient and Positive Exponent
2. 2336 =	2336-2000 or +0336	Positive Coefficient and Positive Exponent
3. 1770 =	1770-1777 or -0007	Positive Coefficient and Negative Exponent
4. 1640 =	1640-1777 or -0137	Positive Coefficient and Negative Exponent
5. 5770 =	5770 2007-2000 or +0007	Negative Coefficient and Positive Exponent
6. 5640 =	5640 2137-2000 or +0137	Negative Coefficient and Positive Exponent
7. 6007 =	6007 1770-1777 or -0007	Negative Coefficient and Negative Exponent
8. 6137 =	6137 1640-1777 or -0137	Negative Coefficient and Negative Exponent

Working these problems will give you practice in floating point numbers.

Directions: Find and list the actual exponent values and the sign of the coefficient and exponent in the blanks provided. Check your answers with those given at the end of the learning activity.

		Actual Value	Sign of Coefficient	Sign of Exponent
1.	5573 =	2204 - 2000 204	Neg	Pos
2.	6414 =	1363 - 1777 -414	Neg	Neg
3.	0023 =	-1777 -1754	Pos	Neg
4.	2732 =	+732	Pos	Pos
5.	0045 =	-1732	Pos	Neg
6.	4444 =	3333 +1333	Neg	Pos
7.	1252 =	-1777 -525	Pos	Neg
8.	3000 =	1777 1000	Pos	Pos
9.	7216 =	0561 - [illegible]16	Neg	Neg

26 AND 27 INSTRUCTIONS

Figure 6-2 shows the relationship between packed and unpacked quantities.

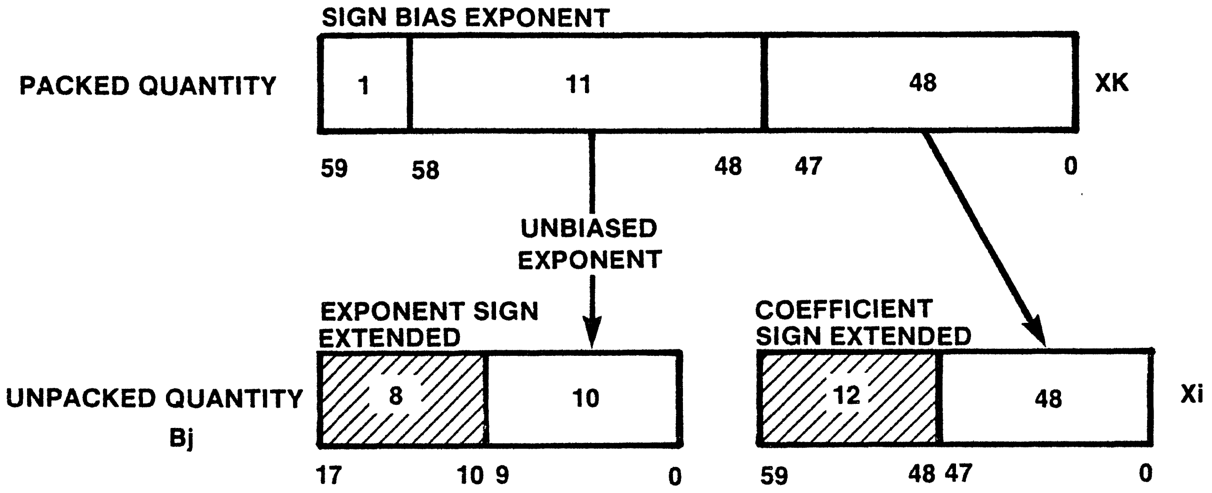


Figure 6-2. CYBER Operand Relationship Packed and Unpacked

The 26 or Unpack instruction reads one operand from Xk. It unpacks the word from floating point format as shown in figure 6-2 and sends the coefficient to Register Xi and the exponent to Register Bj. The 60-bit word sent to Xi consists of the lower 48 bits, bits 0 through 47, of the original operand or Xk. The upper 12 bits, bits 48 through 59, are each equal to the sign of the original operand or bit 59 of Xk. The 18-bit quantity sent to Bj consists of the lower 10 bits, bits 48 through 57, of the unbiased exponent of Xk. The upper eight bits are each equal to the sign of the exponent.

The following shows the sign of the exponent by the combination of bits 59 and 58.

```
59   58          59   58

0 and 0    or    1 and 1 says Negative Exponent
1 and 0    or    0 and 1 says Positive Exponent
```

UNPACKING OPERATION

The following are the basic rules for unpacking a floating point operand.

Basic Machine Method

1. Check bit 59 of operand Xk to determine results to Xi.

    a. If bit 59 is clear (zero), use lower 48 bits of Xk as lower 48 bits of Xi and place zeros in upper 12 bits of Xi.

    b. If bit 59 is set (one), use lower 48 bits of Xk as lower 48 bits of Xi and place ones in upper 12 bits of Xi.

2. To obtain value for Bj Register:

    a. If bit 59 is clear (zero), use bits 48 through 57 of Xk to form Bj bits 0 through 9 and complement bit 58 of Xk, and use this value for bits 10 through 17 of Bj.

    b. If bit 59 is set (one), complement bits 48 through 57 of Xk to form Bj bits 0 through 9 and use bit 58 of Xk, and use this value for bits 10 through 17 of Bj. The following are examples.

    - Bit 59 clear

      Xk = 2002 1234 5676 5432 1000

      Xi = 0000 1234 5676 5432 1000

      Bj = 000002

    - Bit 59 set

      Xk = 4002 1234 5676 5432 1000

      Xi = 7777 1234 5676 5432 1000

      Bj = 001775

Shortcut Method

1. To determine Xi register: use method as shown previously.

2. To obtain value for Bj register:

   a. If bit 59 is clear (zero) and if the exponent is between 2000 and 3777, subtract 2000 from this value, use this value for bits 0 through 9 of Bj, and place zeros in the bits 10 through 17 of Bj.

   b. If bit 59 is clear and if the exponent is between 1777 and 0000, use this value for bits 0 through 9 of Bj and bits 10 through 17 of Bj are set to ones. The following are examples.

   - Bit 59 clear

     Xk = 2002 1234 5676 5432 1000

     Xi = 0000 1234 5676 5432 1000

     $$\begin{array}{r}2002\\ -\underline{2000}\\ 0002\end{array}$$ so Bj = 000002

   - Bit 59 clear

     Xk = 1765 1234 5676 5432 1000

     Xi = 0000 1234 5676 5432 1000

     Bj = 777765

c. If bit 59 is set (one), complement bits 48 through 59 of Xk.

d. If bit 59 is set (one) and if the exponent is between 2000 and 3777 subtract 2000 from this value, use this value for bits 0 through 9 of Bj, and place zeros in bits 10 through 17 of Bj.

e. If bit 59 is set (one) and if the exponent is between 1777 and 0000, use the value for bits 0 through 9 of Bj and bits 10 through 17 of Bj are set to ones.

- Bit 59 set

  Xk = 5765 1234 5676 5432 1000

  Xi = 7777 1234 5676 5432 1000

  $$5765 \xrightarrow{e} \begin{array}{r} 2012 \\ -\underline{2000} \\ 0012 \end{array}$$

  so Bj = 000012

- Bit 59 set

  Xk = 6012 1234 5676 5432 1000

  Xi = 7777 1234 5676 5432 1000

  $6012 \xrightarrow{e} 1765$

  Bj = 777765

The following four examples illustrate the four possible combinations of exponent signs and coefficient signs.

- (Xk) = 2034 4500 3333 2000 0077

  Bit 59 = 0 so coefficient is positive
  Bit 58 = 1 so exponent is positive because bit 59 is not equal to bit 58

  ```
   2034
  -2000
   ────
   0034 so Bj = 000034

           Xi = 0000 4500 3333 2000 0077
  ```

- (Xk) = 1743 4500 3333 2000 0077

  Bit 59 = 0 so coefficient is positive
  Bit 58 = 0 so exponent is negative because bit 59 is equal to bit 58

  1743 so Bj = 777743

  Xi = 0000 4500 3333 2000 0077

- 5743 3277 4444 5777 7700

  Bit 59 = 1 so coefficient is negative
  Bit 58 = 0 so exponent is positive because bit 59 is not equal to bit 58

  5743 is complemented to 2034

  ```
   2034
  -2000
   ────
   0034 so Bj = 000034

           Xi = 7777 3277 4444 5777 7700
  ```

- 6034 3277 4444 5777 7700

  Bit 59 = 1 so coefficient is negative
  Bit 58 = 1 so exponent is negative because bit 59 is equal to bit 58

  6034 is complemented to 1743

  1743 so Bj = 777743

  Xi = 7777 3277 4444 5777 7700

Working these problems will give you practice in unpacking floating point numbers.

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

10. Initial conditions:

```
(X0) = 1732  5644  3215  4444  2222
(X2) = 0020  3761  0000  0000  1111
(X4) = 5020  0000  0000  0000  0002
(X6) = 7626  0432  2222  2222  2222
```

Location 10000 = 26110 26332 26554 26776

What are the contents of the following registers after execution of the above four instructions?

a. X1 = 0000 5644 3215 4444 2222

B1 = 777732

b. X3 = 0000 3761 0000 0000 1111

B3 = 176020

X5 = 7777 0000 0000 0000 0002

c. B5 = 0757

D. X7 = 7777 0432 2222 2222 2222

B7 = 776151

c. X5 =

B5 =

d. X7 =

B7 =

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100 section four for more information.

PACKING OPERATION 27

Basic Machine Method

1. The coefficient is obtained from Xk bits 0 through 47. They become bits 0 through 47 of Xi.

   Bits 48 through 59 of Xk are ignored.

2. The exponent is obtained from Bj bits 0 through 10. Bits 11 through 17 are ignored.

   a. If Xk is positive bit 59 clear (zero), the packed exponent occupying positions bits 48 through 58 of Xi is obtained from bits 0 through 10 of Bj by complementing bit 10.

      Add the sign of the coefficient (bit 59 of Xk).

      Example: Bj = 000003 Xk = 0000 0000 0000 0000 0200

      Xi = 2003 0000 0000 0000 0200

      Example: Bj = 777774 Xk = 0000 0000 0000 0000 0200

      Xi = 1774 0000 0000 0000 0200

   b. If Xk is negative bit 59 is set (one), the packed exponent occupying positions bits 48 through 58 of Xi is obtained from bits 0 through 10 of Bj by complementing bits 0 through 9 and not complementing bit 10.

      Add the sign of the coefficient (bit 59 of (Xk).

      Example: Bj = 000003 Xk = 7777 7777 7777 7777 7577

      Xi = 5774 7777 7777 7777 7577

      Example: Bj = 777774 Xk = 7777 7777 7777 7777 7577

      Xi = 6003 7777 7777 7777 7577

71705

BJ 776072

Short Cut Method

1. The coefficient is obtained from Xk using the method as shown previously.

2. The exponent is obtained from Bj bits 0 through 10, bits 11 through 17 are ignored.

   a. If Xk is positive and Bj is positive, add bits 0 through 9 of Bj to 2000.

   Example: Bj = 000003 Xk = 0000 0000 0000 0200

   ```
    2000
   +0003
   -----
    2003
   ```

   Xi = 2003 0000 0000 0000 0200

   b. If Xk is positive and Bj is negative, subtract the complement of bits 0 through 9 of Bj from 1777.

   Example: Bj = 777774 Xk = 0000 0000 0000 0000 0200

   ```
    1777
   -0003
   -----
    1774
   ```

   Xi = 1774 0000 0000 0000 0200

   c. If Xk is negative and Bj is positive, pack the exponent according to the positive rules and complement the exponent.

   Example: Bj = 000003 Xk = 7777 7777 7777 7777 7577

   ```
    2000
   +0003
   -----
    2003 is complemented to 5774
   ```

   Xi = 5774 7777 7777 7777 7577

   d. If Xk is negative and Bj is negative, pack as above.

   Example: Bj = 777774 Xk = 7777 7777 7777 7777 7577

   ```
    1777
   -0003
   -----
    1774 is complemented to 6003
   ```

   Xi = 6003 7777 7777 7777 7577

The following four examples illustrate the four possible combinations of exponent sign and coefficient.

- (Xk) = 0000 4500 3333 2000 0077
  (Bj) = 000034

  Sign of Xk = 0 so the coefficient is positive
  Sign of Bj = 0 so the exponent is positive

```
 2000
+0034
 ----
 2034 so Xi = 2034  4500  3333  2000  0077
```

- (Xk) = 0000 4500 3333 2000 0077
  (Bj) = 777743

  Sign of Xk = 0 so the coefficient is positive
  Sign of Bj = 1 so the exponent is negative

  777743 is complemented to 000034

```
 1777
-0034
 ----
 1743 so Xi = 1743 4500 3333 2000 0077
```

- (Xk) = 7777 4500 3333 2000 0077
  (Bj) = 000034

  Sign of Xk = 1 so the coefficient is negative
  Sign of Bj = 0 so the exponent is positive

```
 2000
+0034
 ----
 2034 is complemented to  5743
```

  Xi = 5743 4500 3333 2000 0077

- (Xk) = 7777 4500 3333 2000 0077
  (Bj) = 777743

  Sign of Xk = 1 so the coefficient is negative
  Sign of Bj = 1 so the exponent is negative

  777743 is complemented to 000034

```
 1777
-0034
 ----
 1734 is complemented to  6034
```

  Xi = 6034 4500 3333 2000 0077

CYBER 170 Model 750/760 Functional Units
Learning Activity 6-A

Working these problems will give you practice in packing floating point numbers.

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

11. Initial conditions:

```
(X1) = 0000  7777  7777  7777  7777

(B1) = 777735

(X3) = 0000  0000  0000  0000  0001

(B3) = 000073

(X5) = 0000  4000  0000  0000  0000

(B5) = 001000

(X7) = 7777  0000  0000  0000  0000

(B7) = 777776
```

Location 10000 = 27011 27233 27455 27677

What are the contents of the following registers?

a. X0 = 1735 7 — 7

b. X2 = 2073 0000 0000 0000 0001

c. X4 = 2000 4000 0000 0000 0000

2001
4000

d. X6 = 6001 0000 0000 0000 0000

Refer to CYBER 170 (Model 720, 730, 750 and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, section four for more information.

ANSWERS FOR LEARNING ACTIVITY 6-A

	Actual Value	Sign of Coefficient	Sign of Exponent
1.	+204	NEG	POS
2.	-414	NEG	NEG
3.	-1754	POS	NEG
4.	+732	POS	POS
5.	-1732	POS	NEG
6.	+1333	NEG	POS
7.	-525	POS	NEG
8.	+1000	POS	POS
9.	-1216	NEG	NEG

10a. X0 = 1732 5644 3215 4444 2222

Bit 59 = 0 so coefficient is positive
Bit 58 equals Bit 59 so exponent is negative

1732 so B1 = 777732
X1 = 0000 5644 3215 4444 2222

b. X2 = 0020 3761 0000 0000 1111

Bit 59 = 0 so coefficient is positive
Bit 58 equals Bit 59 so exponent is negative

0020 so B3 = 776020
X3 = 0000 3761 0000 0000 1111

ANSWERS FOR LEARNING ACTIVITY 6-A (Contd)

```
c. X4 = 5020  0000  0000  0000  0002

   Bit 59 = 1 so coefficient is negative
   Bit 58 is not equal to Bit 59 so exponent is positive

   5020 is complemented to 2757

   2757
   2000
   ----
   0757 so B5 = 0000757
           X5 = 7777  0000  0000  0000  0002


d. X6 = 7626  0432  2222  2222  2222

   Bit 59 = 1 so coefficient is negative
   Bit 58 is not equal to Bit 59 so exponent is positive

   7626 is complemented to 0151

   0151 so B7 = 776151
           X7 = 7777  0532  2222  2222  2222


11a. X1 = 0000  7777  7777  7777  7777
     B1 = 777735

     Bit 59 of X1 = 1 so coefficient is positive
     Bit 17 of B1 = 1 so exponent is negative

     Exponent = 777735 is complemented to 000042

      1777
     -0042
      ----
      1735

     X0 = 1735  7777  7777  7777  7777
```

ANSWERS FOR LEARNING ACTIVITY 6-A (Contd)

```
b. X3 = 0000  0000  0000  0000  0001
   B3 = 000073

   Bit 59 of X3 = 0 so coefficient is positive
   Bit 17 of B3 = 0 so exponent is positive

   Exponent = 2000
             +0073
             -----
              2073

   X2 = 2073  0000  0000  0000  0000  0001


c. X5 = 0000  4000  0000  0000  0000
   B5 = 001000

   Bit 59 of X5 = 0 so coefficient is positive
   Bit 17 of B5 = 0 so exponent is positive

   Exponent = 2000
             +1000
             -----
              3000

   X4 = 3000  4000  0000  0000  0000


d. X7 = 7777  0000  0000  0000  0000
   B7 = 777776

   Bit 59 of X7 = 1 so coefficient is negative
   Bit 17 of X7 = 1 so exponent is negative

   Exponent = 777776 complement to  000001

    1777
   -0001
   -----
    1776 is complemented to 6001 because of the negative sign
         bits.

   X6 = 6001  0000  0000  0000  0000
```

You have completed learning activity 6-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 6-B. TEXT READING: BOOLEAN PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow used by the 26 and 27 instructions through the Boolean Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the floating point portion of the Boolean Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate B3/B4.

PACK AND UNPACK INSTRUCTIONS

The Boolean instruction requires three clock periods to complete execution. The 26 and 27 instructions require transmissive operations.

INPUT REGISTERS

In quadrant A and C are four input registers. The Xj Register receives all 60 bits of the Xj operand from the CPU. The Xk operand from the CPU goes to two different registers. Bits 48 through 59 go to the Xk Exponent Register and bits 0 through 47 go to the Xk Coefficient Register. The contents of the Bj Register are received from the Bj Register in the CPU. These are all clear/enter type registers. New data is entered every clock period regardless of the instruction being executed. These operands can then be used in the Boolean Unit the following clock period.

CONTROL

In quadrant A is the Instruction Control Translator. It receives f0, m0, m1, and m2 from the CIW Register each clock period. The translator determines the type of operation and selects the proper data paths for that operation.

In quadrant B, GO BOOLEAN from the CPU gates the Xi results through two AND NOT functions. One goes to Xi bits 48 through 59 and the other goes to bits 0 through 47. In quadrant A, GO BOOLEAN gates the

Bj results through two other AND NOT functions. One goes to Bj bits 0 through 11 and the other is EXTEND Bj sign bits 12 through 17. The following control signals are generated by Instruction Control Translator for Pack and Unpack instructions.

Signal	Instruction
EXTEND X	26 · Xk bit 59
GATE B	27
COMP B	27 · Xk bit 59
GATE XK LOWER	26 and 27

The following is an explanation of what the above control signals do.

EXTEND X	Extends the coefficient sign bits for an Unpack (26) instruction when the sign of Xk is negative.
GATE B	Gates bits 0 through 10 of the Bj operand into the complement control circuit for the Pack (27) instruction.
COMP B	Complements the biased exponent for Pack (27) instruction if the sign of the Xk operand is negative.
GATE Xk LOWER	Gates bits 0 through 47 of the Xk operand into the equivalence Circuit for both Pack (27) and Unpack (26) instructions.

METHOD

If GO BOOLEAN sets, then a Boolean instruction was issued during the previous clock period. The data in the Xk and Bj Registers is the operand as described by the j and k designators in that instruction. Data in the unit Input Registers is merged in a static network for transmission to the destination B and/or X Registers. The type of operation and the selection of the data paths is set by Instruction Control Translator.

UNPACK

The Boolean Unit unpacks the floating point quantity from the Xk Operand Register in the CPU. In quadrant A the Xk Exponent Register holds bits 48 through 59 and is the biased exponent. The coefficient (bits 0 through 47) is held in the Xk Coefficient Register located in quadrant C. The exponent's bias is removed in the Unpack Exponent circuits. The sign bit of the exponent is extended to completely fill the 18-bit Bj Register. This data is complemented and is gated by GO BOOLEAN through two AND NOT functions thus storing the true value of the unbiased exponent to the Bj Register.

Xk Coefficient bits 0 through 47 are not complemented, but follow the same logic path as the Transmit Xk. GATE Xk lower controls this part of the operation. EXTEND X causes the coefficient sign to be extended to bits 48 through 59. This completes the 60-bit result to the Xi register.

PACK

The Boolean Unit packs a floating point number to Operand Register Xi. The coefficient of the number is obtained from Operand Register Xk and the exponent from Index Register Bj. The Boolean Unit adds bias to the exponent before combining it with the Xk operand.

The packed exponent (bits 48 through 58 of Xi) or the Result Register is received from the Bj Register. These are bits 0 through 10. If the coefficient is positive, bit 10 is complemented. If the coefficient is negative, bit 10 is not complemented but bits 0 through 9 are. COMP. B, GATE B, and GATE Xk lower signals control the Pack operation.

You have completed learning activity 6-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 6-C. TEXT READING: BOOLEAN DETAILED PAK DIAGRAM

This activity follows the data bits and control flow used by the 26 and 27 instructions through the Boolean Unit detailed pak 3.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the floating point portion of the Boolean Unit detailed 3.0 diagram.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate C1/C2.

INPUT REGISTERS

For floating point operations the unit uses only three of the four input registers. In quadrant A is a KM module at 6C15. This module has the Xk exponent bits 48 through 59. The KK module below it at 6C11 has the Bj Register. This receives the unpacked or unbiased exponent from the Bj Register in the CPU. In quadrant C there are four KK modules (location 6C07 through 6C10). These contain the Xk coefficient bits 0 through 47. Data is loaded to these registers every clock period no matter what instruction is being executed. On the next clock period the Pack or Unpack instruction can be executed.

CONTROL

The instruction translation is done on the KM module at 6C15 in quadrant A. It receives f0, m0, m1, and m2 every clock period from the CIW in the CPU. These translations determine the function and the data paths required by the instruction. The chart on 6C15 shows which gates are used for the different instructions. The GO BOOLEAN is received on the second clock period from the CPU and is all that is necessary to transmit the results to Xi and, if there is a 26 instruction, to Bj also.

METHOD

Unpack 26 Instruction

This operation requires five separate data paths, one for the exponent going to the Bj Register bits 0 through 9, EXTEND Bj sign bits 10 and 11, EXTEND Bj sign bits 12 through 17, coefficient bits 0 through 47, and Xk sign EXTEND bits 48 through 59 both going to the Xi Register.

Unpacking of the exponent takes place on the KM module at 7C15 in quadrant A. In learning activity 6-A, one of the examples used for unpacking was Xk = 6012 1234 5676 1000. Taking this number through these circuits gives these results: bits 48 through 58 go to the Odd Clock Delay. Bits 48 through 57 go to Complement Control. Since Xk bit 59 is set and is complemented, Xk bits 48 through 57 or Bj bits 0 through 9 will not be complemented in Complement Control. When GO BOOLEAN equals one, these bits are complemented as transmitted to Bj through an AND NOT function, so bits 0 through 9 equal 1765. Xk bit 58 goes from the Odd Clock Delay to the adder where a Logical Difference is done against Xk bit 59. In this example both bits 58 and 59 equaled one, so the difference would be zero. This will be complemented and gated by GO BOOLEAN through another AND NOT function. Bj bits 10 and 11 will equal ones. Bj now equals 7765. In quadrant B the KN module at 6C16 (the upper portion) generates the upper six bits (12 through 17) of the Bj Register. Again you do a Logical Difference on Xk bits 58 and 59, and this would be equal to zero. This would be complemented and gated by GO BOOLEAN through an AND NOT function. Bj bits 12 through 17 now equal ones and the Bj Register holds 777765 or the unbiased exponent. These upper bits tell that the exponent was negative. B Register Access Control prevents entry to the destination B Register from the Boolean Unit for all instructions except Unpack 26 instruction.

The coefficient or Xk bits 0 through 47 leave the KK modules in quadrant C and go to the KL modules in quadrant D. COMP Xk is not enabled for a 26 instruction so these bits are not complemented. They pass through the Equivalence Circuit when gated by GATE Xk Lower is enabled. Because GATE Xj is not enabled, the output of the Equivalence Circuit is the complement of the input. This is recomplemented when gated by GO BOOLEAN through an AND NOT function to the Xi Register, so the output matches the input. Bits 48 through 59 of Xk are needed as coefficients sign bits. The KN module with Xk bits 48 through 59 gated to Complement Control is in quadrant B. EXTEND X is enabled because you have a 26 instruction and bit 59 equals one. This is complemented causing a zero on the AND function. This goes down to the AND NOT function causing all ones for bits 48 through 59 to the Xi Register.

The following values are in the Result Registers.

```
    Bj = 777765
    Xi = 7777 1234 5676 5432 1000
```

This agrees with the example.

Pack 27 Instructions

This instruction needs separate data paths from the B and X Registers. With the exponent loaded into the Bj Register on the KK module at 6C11 in quadrant A, bits 0 through 10 go to the KN module at 6C16 in quadrant D. Bit 11 is terminated or discarded because it is an extension of the exponents sign bit (bit 10). Using the example used during unpack, 3765 goes into the Complement Bit 10 to Add Bias circuit giving a 1765 for an output. This is gated by GATE B Complement Control. COMP B will be set equal to one because of the 27 instruction and bit 59. Because of this the data is not complemented. It merges with the Complement output (bit 59) and is equal to a zero. This value of 1765 goes to the AND NOT function. GO BOOLEAN and EXTEND X to the AND function equal ones because you are not doing a 26 instruction. The third input Xi bits 48 through 59, equals one because Equivalence Circuit equals one and bits 48 through 59 equal one as GATE LP equals zero. With all ones in to the AND NOT function the output would be 6012 for bits 48 through 59 of Xi. Packing of the coefficient is a straightforward transmission of bits 0 through 47. In quadrant C these bits are received from the CPU. They are sent to the KL modules 6C12, 13, and 14 in quadrant D. GATE Xk Lower is set equal to a one so the gate is enabled to the Equivalence Circuit. Since Gate Xj is not set, the output is the complement of the input. With GO BOOLEAN to the AND NOT function, true data is sent to bits 0 through 47 of Xi. This gives you the following data in Xi.

```
    Xi = 6012 1234 5676 1000
```

You have completed learning activity 6-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 6-D. EXERCISE: BOOLEAN TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the floating point portion of the Boolean Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W number 53 (2 of 10) in your microfiche viewer and select coordinate C1/C2.

CYBER 170 Model 750/760 Functional Units
Learning Activity 6-D

Working these problems will give you practice in troubleshooting problems in floating point portion of the Boolean Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the module(s) location and if possible the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction executed was a 27123.

```
Given:     X3 = 7777 7777 7777 7777 7776
           B2 = 000060
           X1 = 7777 7777 7777 7777 7776
```

6C15

2. Instruction executed was a 27123.

```
Given:     X3 = 0000 0000 0000 0000 0004
           B2 = 000060
           X1 = 5717 0000 0000 0000 0004
```

6C/5

3. Instruction executed was a 26123.

```
Given:     X3 = 5777 6777 7777 7777 7777
           B2 = 000000
           X1 = 0000 6777 7777 7777 7777
```

6C/5

4. Instruction executed was a 26123.

```
Given:    X3 = 1000 4000 0000 0000 0000
          B2 = 007000
          X1 = 0000 4000 0000 0000 0000
```

6016

The answers to the previous exercise are on the following page.

ANSWERS FOR LEARNING ACTIVITY 6-D

In the answers to these problems we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting we have included them.

1. The correct answer is:

   X1 = 5717 7777 7777 7777 7776

   Possible malfunction(s):

   a. 6C15 - Extend X signal KM module

   b. 6C16 - No gate output KN module

   c. 6C15 - Gate B signal KM module

   d. 6C15 - Gate Xk upper KM module

2. The correct answer is:

   X1 = 2060 0000 0000 0000 0004

   Possible malfunction(s):

   a. 6C15 - Complement B term KM module

3. The correct answer is:

   X1 = 7777 6777 7777 7777 7777
   B2 = 000000

   Possible malfunction(s):

   a. 6C15 - Extend X signal KM module

4. The correct answer is:

   X1 = 0000 4000 0000 0000 0000
   B2 = 777000

   Possible malfunction(s):

   a. 6C16 - Extend Bj signal KN module

MODULE 7
NORMALIZE UNIT

The module 7 learning activities will acquaint you with the logical operations of the Normalize Functional Unit and review the function of the instruction.

PRETEST

Before you begin this module, sign on the PLATO terminal and take the module 7 test. Do not waste time on answers you do not know.

By taking the test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column below, put a check mark by each activity assigned to you by PLM and then proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities, or do some activities more than once.

Assigned	Completed	Activity	Description	Page
_____	_____	7-A	Programmed Text: Normalize Instruction. This activity reviews the function of the Normalize instruction.	7-3
_____	_____	7-B	Text Reading: Normalize Primary Block Diagram. This activity traces the data bits and control flow through the 1.0 diagram.	7-18
_____	_____	7-C	Text Reading: Normalize Detailed Pak Diagram. This activity traces the data bits and control flow through the 3.0 diagram.	7-22
_____	_____	7-D	Exercise: Normalize Trouble shooting. This activity allows you to develop some basic methods of troubleshooting the Normalize Functional Unit.	7-28

LEARNING ACTIVITY 7-A. PROGRAMMED TEXT: NORMALIZE INSTRUCTION

This activity reviews the function of the Normalize instruction. You will also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 24 and 25 instructions.

NORMALIZE INSTRUCTION

The 24 or Normalize instruction causes the Normalize Unit to read one operand from the Xk Register, performs a normalizing operation on this word in a floating point format, and delivers the normalized result to the Xi Register. In addition, the unit delivers a positive integer shift count to the Bj Register. This shift count is the number of bit positions of shift required to normalize the original operand coefficient. The normalize operation is the repositioning of the coefficient portion of the operand, then adjusting the exponent of the operand so that the actual value has not changed. The coefficient is shifted left toward the highest order bit. It is shifted left until bit 47 is different, that is, not equal to the coefficient sign bit 59. This places the most significant bit of the coefficient in the highest order bit position. The exponent is then decreased by the number of bit positions shifted. For more information, read the instructions description in section four of the CYBER 170 (Model 720, 730, 750, and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100. For further study, read Special Case Operands in section five of the same manual.

The following are the basic rules for Normalizing a floating point number.

- If bit 59 of Xk Register is set (one), complement all 60 bits at the start, Normalize, then recomplement before sending the results to Xi.

- Normalizing consists of left shifting the coefficient the minimum number of positions required to make bit 47 different than bit 59. This places the most significant bit of the coefficient in the highest order position of the coefficient.

- If the adjusted unpack exponent is equal to -1777 the operation will proceed as normal.

- If the adjusted unpack exponent is more negative than -1777, a zero word underflow is sent to the Xi Register. The shift count is sent to the Bj Register.

- Normalizing either plus or minus zero coefficients will set the Bj Register equal to 60_8 and the Xi Register will be equal to all zero bits.

The following are two examples of the Normalize operation.

- Xk = 2034 0047 6500 0000 2262

  Need a LS of six to make bit 47 not equal to bit 59

  Unpack exponent and compute new Unpack value

  ```
    2034
    2000
  + 34
  -  6  Number of shifts to Normalize
  + 26
  ```

  Repack Exponent

  ```
  2000
    26
  2026
  ```

  Results

  Xi = 2026 4765 0000 0022 6200

  Bj = 000006

- Xk = 5743 7730 1277 7777 5515

  Complement because bit 59 = 1

  2034 0047 6500 0000 2262

  Need an LS of six to make bit 47 not equal to bit 59

  Unpack exponent and compute new Unpack value

```
 2034
 2000
+  34
-   6
+  26
```

  Repack Exponent

```
 2000
+  26
 2026
```

  Results

  Xi = 2026 4765 0000 0022 6200

  Recomplement as you complemented at start

  Xi = 5751 3012 7777 7755 1577

  Bj = 000006

This exercise tests your understanding of the Normalize operation.

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

1. Initial conditions:

   (X0) = 3770 3777 3777 3777 3777

   (X2) = 5774 7737 7777 7777 7777

   (X4) = 3260 0000 0004 0000 0000

   (X6) = 5517 7777 3777 7777 7773

   Location 10000 = 24110 24332 24554 24776

   What are the contents of the following registers after execution of the preceding four instructions?

   a. 0 0 0 0
   1770
   -1
   1767

   X1 = 3767 7776 7776 7776 7776

   B1 = 00001

   b.
   0003 0040 0000 0000000000
   -1000
   0000

   -6

   X3 = 6003 3777 7777 777777777777

   B3 = 000006

   1777
   3
   1774

   6003 3

c. 3260 0000 0004 0000 0000
-2000
1260
-25
1233

X5 = 3233 4000 0000 0000 0000

B5 = 000025

5517 7777 3777 7777 7773

d. 2260 0000 4000 0000 0004
-2000
260
-14
2244
5533

X7 = 3777 7777 7773 7777

B7 = -14

2. Normalize X3 = 1777 0000 0000 0000 2222. The instruction was 24123.

+ indefinite

Refer to CYBER 170 (Model 720, 730, 750, and 760) Model 176, (Level B), Hardware Reference Manual, publication number 60456100, for more information.

ROUND NORMALIZE INSTRUCTION

The 25 or Round Normalize instruction is the same as the Normalize, except the rounding operation consists of adding a bit to the coefficient portion of the Xk operand. This bit position is 2^{-1} (as shown in table 6-1) of the noncoefficient. This bit is always the complement of the coefficients sign bit 59. This results in increasing the value of the least significant bit of one half. This round bit is shifted into the coefficient when left shifting to Normalize. All rules that apply to Normalize apply to Round Normalize.

The following are two examples of the Round Normalize operation.

- Xk = 2034 0047 6500 0000 2262

  Place round bit at 2^{-1} and not equal to bit 59

  2034 0047 6500 0000 2262 4

  Need a LS of 6 to Normalize

  Unpack exponent and compute new value

  2034

  2000

  + 34

  - 6

  + 26

  Repack Exponent

  2000
  26
  2026

  Results

  Xi = 2026 4765 0000 0022 6240

  Bj = 000006

- Xk = 5743 7730 1277 7777 5515

  Complement because bit 59 = 1 to

  2034 0047 6500 0000 2262

  Place round bit 2^{-1} and not equal to bit 59

  2034 0047 6500 0000 $2262_{\wedge}4$

  Need a LS of six to Normalize

  Unpack exponent and compute new value

```
 2034
 2000
+  34
-   6
+  26
```

  Repack Exponent

```
2000
  26
2026
```

  Results

  Xi = 2026 4765 0000 0022 6240

  Recomplement as you complemented at start to

  Xi = 5751 3012 7777 7755 1537

  Bj = 000006

Working this problem will test your understanding of the Round Normalize Operation.

Directions: Work the following problems. Check your answers with those given at the end of the learning activity.

3. Initial conditions:

(X0) = 3770 3777 3777 3777 3777

(X2) = 5517 7777 3777 7777 7773

Location 10000 = 25110 25332 46000 46000

What are the contents of the following registers after execution of the preceding two instructions?

a. 3770 3777 3777 3777 3777 4
2000
1770 7776 7776 7776 7777
— 1

X1 = 3767 7776 7776 7776 7777

B1 = 000001

b. 5517 7777 3777 7777 7773
2260 0006 4000 0000 5004 4
− 2000
14

X3 = 5533 3777 7777 7773 3777

B3 = 000014

2244 4000 0000 0004 4000
5533 3777 7777 7773 3777

ANSWERS FOR LEARNING ACTIVITY 7-A

1a. X0 = 3770 3777 3777 3777 3777

Need an LS of 1

Unpack Exponent

```
 3770
-2000
+1770
-   1 Number of shifts
+1767
```

Repack Exponent

```
 2000
+1767
+3767
```

Result

X1 = 3767 7776 7776 7776 7776

B1 = 000001

ANSWERS FOR LEARNING ACTIVITY 7-A (Contd)

b. X2 = 5774 7737 7777 7777 7777

Complement because bit 59 = 1 to

2003 0040 0000 0000 0000

Need a LS of 6

Unpack Exponent

```
 2003
-2000
+   3
-   6  Number of shifts
-   3  Note that you now have a negative
       exponent.
```

Repack Exponent

```
 1777
-   3
 1774
```

Result

X3 = 1774 4000 0000 0000 0000

Recomplement as you complemented at the start to

X3 = 6003 3777 7777 7777 7777

B3 = 000006

ANSWERS FOR LEARNING ACTIVITY 7-A (Contd)

c. X4 = 3260 0000 0004 0000 0000

Need an LS of 25

Unpack Exponent

```
 3260
-2000
+1260
-  25
 1233
```

Repack Exponent

```
 2000
+1233
 3233
```

Result

X5 = 3233 4000 0000 0000 0000

B5 = 000025

ANSWERS FOR LEARNING ACTIVITY 7-A (Contd)

1d. X6 = 5517 7777 3777 7777 7773

Complement because bit 59 = 1 to

2260 0000 4000 0000 0000

Need a LS of 14

Unpack Exponent

```
 2260
-2000
+ 260
-  14
+ 244
```

Repack Exponent

```
 2000
+0244
 2244
```

Result

X7 = 2244 4000 0000 0000 0000

Recomplement as you complemented at the start to

X7 = 5733 3777 7777 7773 7777

B7 = 000014

2. Special Situation.

Have 1777 or + indefinite so copy Xk to Xi Register and set Bj to all zeros. Also no shifting is done.

Result

X1 = 1777 0000 0000 0000 2222

B2 = 000000

ANSWERS FOR LEARNING ACTIVITY 7-A (Contd)

3a. X0 = 3770 3777 3777 3777 3777

Place round bit at 2^{-1} and not equal to bit 59

3770 3777 3777 3777 3777∧4

Need an LS of 1

Unpack Exponent

```
 3770
-2000
+1770
-   1  Number of shifts
+1767
```

Repack Exponent

```
 2000
+1767
 3767
```

Result

X1 = 3767 7776 7776 7776 7777

B1 = 000001

ANSWERS FOR LEARNING ACTIVITY 7-A (Contd)

b. X2 = 5517 7777 3777 7777 7773

Complement because bit 59 = 1 to

2260 0000 4000 0000 0004

Place round bit at 2^{-1} and not equal to bit 59

2260 0000 4000 0000 0004∧4

Need an LS of 14

Unpack Exponent

```
 2260
 2000
+0260
-  14  Number of shifts
+0244
```

Repack Exponent

```
 2000
+ 244
 2244
```

Result

X3 = 2244 4000 0000 0004 4000

Recomplement as you complemented at the start to

X3 = 5533 3777 7777 7773 3777

B3 = 000014

You have completed learning activity 7-A. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 7-B. TEXT READING: NORMALIZE PRIMARY BLOCK DIAGRAM

This activity follows the data bits and control flow used by the 24 and 25 instructions through the Normalize Functional Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Normalize Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 54 (3 of 10) in your microfiche viewer and select coordinate E3/E4.

NORMALIZE INSTRUCTION

The Normalize instruction requires three clock periods to complete execution. The 24 and 25 instructions are identical except the 25 instruction adds the round bit to the coefficient. The Normalize Unit operates on only positive operands. Therefore it complements negative operands before operating on them.

INPUT REGISTERS

In quadrant A is the Xk Exponent Input Register. This receives bits 48 through 57 of the Xk operand from the CPU. In quadrant C is the Xk Coefficient Input Register. This receives bits 0 through 47 of the Xk operand. These are also clear/enter type registers. The data is held for use until the second clock period. In quadrant B a register holds the partial difference of the shift count from the exponent. This data is used during the third clock period. In quadrant D are two other registers. Decode Shift Count Register holds the shift count. The other holds the coefficient after being shifted by Shift Rank 1. The data from both of these registers is further used during the third clock period.

CONTROL

Following is a list of control signals and their functions.

Quadrant A

ROUND This is actually m0. It enters during the clock period that the CIW Register issues an instruction. It is used as a gating term.

COMP if Xk Neg This complements Xk exponent if the coefficient is negative. It causes the subtraction of the shift count from the exponent and gates the shift count to the shift ranks.

Quadrant B

EXTEND Bj The unit sends the 6-bit shift count to the Bj Register and EXTEND Bj sends zeros to the rest of the Bj Register bits 6 through 17.

GATE OUTPUT This gates the exponent and coefficient from the unit to the result or Xi Register.

GO NORMALIZE This is received in the clock period after the 24 or 25 instruction was issued by the CIW Register. It causes EXTEND Bj and GATE OUTPUT control signals.

Quadrant C

COMP if Xk Neg This complements Xk coefficient if coefficient is negative.

m0 Control signal m0 is received during the same clock period the instruction was issued from CIW Register. It has no significance unless the GO NORMALIZE Flag is set.

ROUND — This causes the addition of the round bit in Shift Rank 1 if the shift count is odd. It is generated by m0 being set.

SHIFT LEFT ONE POSITION — This causes a bit to be shifted left one place in Rank 1.

Quadrant D

$\overline{\text{SHIFT}}$ COUNT Bit-0 — This causes generation of Round bit from Round Flag-2.

ROUND BIT — This causes the addition of the Round bit in Shift Ranks 2, 3, or 4 if the shift count is even.

LEFT SHIFT 0+2 — This causes a bit to be shifted left two places or no places in Rank 2.

LEFT SHIFT 0+4+8+12 — This causes a bit to be shifted left four or eight or 12 or no places in Rank 3.

LEFT SHIFT 0+16+32+64 — This causes a bit to be shifted left 16 or 32 or 64 or no places in Rank 4.

There is one other control signal but is not shown on the 1.0 diagrams.

NORMALIZE UNDERFLOW — If a complete unflow occurs, this will cause the one bits of the underflow word to be sent to the Xi Register as zero bits.

METHOD

The data is received in the Exponent and Coefficient Input Register in quadrants A and C during the first of the three clock periods. At the same time the instruction was issued from the CIW Register. During the second clock period, the exponent in quadrant A is taken from the Exponent Input Register. If bit 59 equals one then the coefficient is negative, so it will be complemented. The exponent is again complemented going into the Exponent Shift Count First Stage Adder. The second input is from the Static Shift Count Determination Network in quadrant C. This network has determined the amount of left shifts necessary to normalize. This data is gated by ONES CHECK. From the First Stage Adder it goes to the Exponent Shift Count Second Stage Adder in quadrant B. This is then gated to the Xi Register by Gate Output through an AND NOT function during the third clock period.

If doing a ROUND and the shift count is odd the round bit is entered into the Shift Rank 1 in quadrant C. If shift count is even then the round bit will be entered in either Shift Rank 2, 3, or 4 in quadrant D.

You have completed learning activity 7-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 7-C. TEXT READING: NORMALIZE DETAILED PAK DIAGRAM

This activity follows the data bits and control flow used by the 24 and 25 instructions through the Normalize Unit detailed pak 3.0 diagram.

OBJECTIVE

- You will be able to follow the data bits and control flow through the Normalize Unit detailed 3.0 diagrams.

Directions: Place microfiche GF60420300W number 54 (3 of 10) in your microfiche viewer and select coordinate F1/F2.

INPUT REGISTERS

In quadrant A you have an EE module at 7J14. It receives the coefficient sign bit to the bit 59 Normalize Sign Register and the packed or biased exponent bits 48 through 58 to the Exponent Input Register.

In quadrant C are three EA modules (7J02 through 7J04). Each module receives 16 bits of the coefficient and also the coefficient's sign bit.

CONTROLS FOR NORMAL OPERATIONS

In quadrant A and the EE module (at the bottom) is m0. If doing a 25 or Round Normalize, this would be set. On the EE module is the Enable Shift signal. It gates your shift count to the first stage of the adder on the EF module in quadrant A and B. It gates the shift count in quadrant C on the EG module at 7J15. There are three EC modules at 7J05 through 7J07. ENABLE SHIFT Gates Shift Count bit 0 to the Shift Rank 1 and allows a left shift of one. In quadrant A is an EF module and GO NORMALIZE from the CPU. This is used on the third clock period to gate the results to the Xi and Bj Registers.

METHOD

The following is an example of this operation divided into the three clock periods required to execute the instruction using a 24 or Normalize instruction with no Special Case and a Floating Point value of 5517 7777 3777 7777 7777.

First Clock Period: With the exponent in quadrant A and the EE module at 7J14, the coefficients sign bit is stored in a Bit 59 Normalize Sign Register. The remaining 11 bits of the biased exponent are put in the Exponent Input Register. In quadrant C each EA module at 7J02 through 7J04 will store 16 bits of the coefficient in the Coefficient Input Register and the coefficient's sign bit in the bit 59 Normalized Sign Register.

Second Clock Period: With the exponent in quadrant A and the EE module, the example bit 59 was set. Using Note 1 in quadrant D, bits 48 through 59 are now bits 0 through 10 and are complemented in Complement Control. Bit 59 is also referred to as X Sign. It goes to the CPU X Register sign control. This controls the complementing of the result at the input of the destination or Xi Register. Bits 0 through 10 go to the EF module at 7J16. Taking the path of bits 0 through 5, it is complemented and goes to the Exponent minus the Shift Count First Stage. The other input gated by Enable Shift to the adder is Shift Count bits 0 through 5. This is determined by the Static Shift Count Determination Network on the EB module at 7J12 in quadrant C. The results of the subtraction are stored in the Bit Enables and Bit Borrow Registers. The remaining 5 bits 6 through 10 that entered the EF module are complemented and stored in the Bit Enable Register.

In quadrant C and the EA modules, the coefficient will be complemented because bit 59 equals a one. Using the top data path it goes to the EB module where the shift count is determined. Shift Count bits 0 through 5 go to the EF module, which has been already discussed and to an EG module at 7J15. These bits are gated by Enable Shift through the AND NOT function to three different registers. Shift Count Registers bits 0 through 5 should equal 63. Bit 1 Register should equal one. And the Decode Shift Count Register bits 2 and 3 equal a shift of 12 decimal. A quick check for what shift counts will be used can be made by using table 7-1.

TABLE 7-1. SHIFTS PER/SHIFT COUNT BITS

32	16	8	4	2	1	Shifts
5	4	3	2	1	0	Shift Count Bits

In this example, the coefficient had to be shifted 12_{10} or 14_8 places. The shift count is in octal, so shift count bits 3 and 2 would be set. Bit 3 shifts 8_{10} places and bit 2 causes a shift of 4_{10} places for a total of 12_{10} places. The last logic used during the second clock are the three EC modules 7J05 through 7J07. These modules do a left shift of one if requested. Shift Count bit 0 is gated by Enable Shift. This corresponds to bit 0 and equals a shift of one in table 7-1. Since the example used does not use a shift of one the data in the register will be unchanged.

Third Clock Period: The exponent is in quadrant B. The partial difference in the EF module Registers is sent to the EH module at 7J13. This completes the difference on all 11 bits of the exponent in the exponent minus the Shift count Second Stage Adder. The result is gated by GATE OUTPUT. This was generated by GO NORMALIZE on the EG module through an AND NOT function. This would give a 2244 going to the Xi Register in the CPU. As for the coefficient, the coefficient data will leave the EC modules in quadrant C to the ED module at 7J08 through 6J11 in quadrant D. The ranks used to shift the coefficient are controlled by the shift count bits from the EG modules Bit 1 Register and the Decoded Shift Count Register.

Since it is a shift of 12_{10} Ranks 2 and 4 will not be used to shift the coefficient. The output of the shift ranks is gated through an AND NOT function by GATE OUTPUT. A value of 4000 0000 0000 0000 goes to the Xi Register in the CPU. Since bit 59 was set, X Sign was set and the whole value of 2244 4000 0000 0000 0000 will be complemented to 5533 3777 7777 7777 7777. This example was the same as example 3b except that example was a round instruction.

In the examples worked, the shift count was always sent to a Bj Register. This takes place during the third clock period. In quadrant C on the EG module, the shift count was stored in the Shift Count Register and was equal to 63. EXTEND Bj generated by GO NORMALIZE will gate this data through an AND NOT function. This value is equal to 14 or bits 0 through 5 go to the Bj Register in the CPU. The EXTEND Bj command equals one. It is complemented through a Fan Out making bits 6 through 11 equal to zeros. The remaining 6 bits needed by Bj are generated by EXTEND Bj on the EH module at 7J13 in quadrant B. They will be bits 12 through 17 of Bj. So in our example Bj equals 000014.

If the coefficient is equal to all zeros, a 24 instruction would be impossible. The unit sends all zeros to Xi and a 60_8 count to Bj. The all zeros to Xi is caused by blocking GATE OUTPUT. This is done on the EF modules adder in quadrant A. With a Shift count 48, ENABLE SHIFT and NOT ROUND would cause all ones at the AND function. This would be held in the register and on the third clock it would be complemented breaking the next AND function causing no GATE OUTPUT on the EH and ED modules in quadrants B and D. The Shift Count Determination Network on the EB module in quadrant C generates a shift count of 60_8 or 48_{10}. This goes to the EG module and is sent to the Bj Register as explained previously.

A 25 or Round Normalize instruction takes place in the following manner. In quadrant A and the EE module m0 is set and sets ROUND Flag. This generates ROUND which goes to the Shift Rank logic. Taking the EC modules in quadrant C first, with an uneven shift count the round bit will enter the EC module. With an even shift count it would enter on the ED modules in quadrant D. Going back to the EE module in quadrant A, the ROUND signal goes to an EF module. This module set ROUND Flag 2. This generates a round bit and goes to either ED modules at 7J09 or 7J11. The shift count determines which of the two modules brings in the round bit. If ROUND bit equals 2^{-1}, for instance, shift count equals 2 and ROUND bit is now bit 2^1 on 7J09. If shift count equals 4, ROUND bit is now bit 2^3 on 7J11. This pattern continues for all even shift counts. In any case, the round bit is shifted along with the other bits of the coefficient. This completes valid operatons of the Normalize Unit.

SPECIAL CASES

Reviewing the three possible special cases with floating point numbers, make use of table 6-2, Floating Point Numbers.

Overflow: Overflow is a positive exponent with an actual value of +1777 (or 3777 or 4000 in packed/biased form). This is the largest exponent value that can be represented. This exponent value may result from a calculation along with the coefficient that represents a correct result. This situation is called partial overflow. Further use of this result will generate complete overflow. A complete overflow occurs when the result is greater than a positive 1777. Overflow is also called positive infinity ($+\infty$) and negative infinity ($-\infty$).

Underflow: Underflow is a negative exponent with an actual value of -1777 (or 0000 or 7777 in packed/biased form). This is the smallest exponent value that can be represented. This exponent value may result from a calculation along with the coefficient that represents a correct result. This situation is called partial underflow. Further use of this result will generate complete underflow. A complete underflow occurs when the result is less than a negative 1777. Underflow is also called positive underflow (+0) and negative underflow (-0).

Indefinite: An indefinite results whenever the calculation cannot be resolved. The indefinite result is a value that cannot occur in normal floating point calculations, and is a negative exponent with a value of -0000 (or 1777 or 6000 in packed/bias form). Indefinite is also called positive indefinite (+Ind) and negative indefinite (-Ind).

Logic Diagrams for Special Cases

Method

In quadrant A and the EE module, the $\overline{\text{Ones}}$ Check portion checks the exponent for Overflow 3777 or Indefinite 1777 after it has been put in a positive condition. If either of these cases exist, the unit will block the shift count, so the operand passes through without change and the value sent to the Bj Register is zero.

The $\overline{\text{Ones}}$ Check logic does a ones check on exponent bits 0 through 9. If they are all equal to ones, either Overflow 3777 or Indefinite 1777 exists. When this happens, there will be no ENABLE SHIFT. With no ENABLE SHIFT to the EG and EC modules in quadrant C, the coefficient will pass through unchanged. On the EF module in quadrant A, ENABLE SHIFT also gates in Shift Count to the adder. With no input the exponent does not change. So Xi is just a copy of Xk.

If Bj equals all zeros and the EG module in quadrant C has no ENABLE SHIFT, the shift count will not be entered into the Shift Count Register bits 0 through 5. When EXTEND Bj comes up bits 0 through 5 will be stored as all zeros. It will be all zeros for bits 6 through 17 as explained previously. If

Xk = 3777 1245 6666 5555 3333 the results in

Xi = 3777 1245 6666 5555 3333 and

Bj = 000000

Because subtracting the shift count from the exponent can cause complete underflow by exceeding its negative range the EF module in quadrant B has the Test Underflow logic. If this condition exists, the unit blocks GATE OUTPUT and causes a zero and all 60 bits equal to zeros to be stored in Xi. The shift count goes to the Bj Register unaltered.

You have completed learning activity 7-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 7-D. EXERCISE: NORMALIZE TROUBLESHOOTING

This activity allows you to develop approaches to troubleshooting the Normalize Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W number 54 (3 or 10) in your microfiche viewer and select coordinate F1/F2.

This exercise gives you practice in troubleshooting problems in the Normalize Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the module(s) location and if possible the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction executed was a 24567.

```
Given:    X7 = 2005 0077 7777 7777 7777
          X5 = 1775 7777 7777 7777 7600
          B6 = 000007
```

4FB7 - 7J12 RIT41

4FA7 - 7J04 BIT41

2. Instruction executed was a 24567.

```
Given:     X7 = 2000 0000 0000 0000 0000
           X5 = 2000 7777 7777 7777 7771
           B6 = 000000
```

3. Instruction executed was a 24567.

```
Given:     X7 = 5777 7777 7777 7777 7771
           X5 = 7777 7777 7777 7777 7777
           B6 = 777777
```

13

4. Instruction executed was a 24567.

```
Given:     X7 = 7774 7600 0000 0000 0000
           X5 = 4001 0000 0000 0000 0037
           B6 = 000005
```

0.00[illegible] 017[illegible]

[illegible]

[illegible] 776[illegible]

7J16

37

~~4 0 A1~~

5. Instruction executed was a 25567.

```
Given:     X7 = 1720 0000 0000 0000 0006
           X5 = 1643 6000 0000 0000 0000
           B6 = 000055
```

ROUND.[illegible]

7J14 4EE7

7J05 4EC7

6. Instruction executed was a 25567.

```
Given:     X7 = 1720 0000 0000 0000 0006
           X5 = 1720 7777 7400 0000 0006
           B6 = 000000
```

7504

ANSWERS FOR LEARNING ACTIVITY 7-D

In the answers to these problems, we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting we have included them.

1. The correct answer is:

   X5 = 1776 7777 7777 7777 7700
   B6 = 000006

   Possible malfunction(s):

   a. 7J12 - Incorrect normalize count EB module TP61 = 0

   b. 7J04 - Incorrect input on EA module TP23 = 1

2. The correct answer is:

   X5 = 1722 6000 0000 0000 0000
   B6 = 000055

   Possible malfunction(s):

   a. 7J14 - Negative sign input EE module pin 52 = 1
   Detects an indefinite or infinite operand

3. The correct answer is:

   X5 = 6055 1777 7777 7777 7777
   B6 = 000055

   Possible malfunction(s):

   a. 7J16 - Go Normalize term blocked on module
   TP52 = 1

ANSWERS FOR LEARNING ACTIVITY 7-D (Contd)

4. The correct answer is:

   X5 = 7777 7777 7777 7777 7777
   B6 = 000005

   Possible malfunction(s)

   a. 7J16 - Underflow Detector on EF module TP63 = 1

5. The correct answer is:

   X5 = 1643 6400 0000 0000 0000
   B6 = 000055

   Possible malfunction(s)

   a. 7J14 - Lost Round Bit on EE module TP65 = 1

   b. 7J05 - Lost Round Bit on EC module TP04 = 1

6. The correct answer is:

   X5 = 1643 6400 0000 0000 0000
   B6 = 000055

   Possible malfunction(s)

   a. 7J04 - Complement gate on EA module 4C term = 1

MODULE 8
FLOATING ADD FUNCTIONAL UNIT

The module 8 learning activities acquaint you with the logical operations of the Long Add Functional Unit. Review the function of the instruction.

PRETEST

Before you begin this module, sign on the PLATO terminal and take the module 8 test. Do not waste time on answers you do not know. By taking the test, you may be able to skip some of the learning activities in this module.

LEARNING ACTIVITIES

In the Assigned column below, put a check mark by each learning activity assigned to you by PLM and proceed through your assigned activities. Check off each activity as you complete it. You may choose to do all of these activities, or do some activities more than once.

Assigned	Completed	Activity*	Description	Page
_____	_____	8-A	Programmed Text: Floating Add Instructions. During this activity you will review the function of the Floating Add instructions.	8-3
_____	_____	8-B	Text Reading: Floating Add Instructions. During this activity you will trace the data bits and control flow through the 1.0 diagrams.	8-35
_____	_____	8-C	Text Reading: Floating Add Detailed Pak Diagrams. During this activity you will trace the data bits and control flow through the 3.0 diagrams.	8-39

CYBER 170 Model 750/760 Functional Units
Module 8

Assigned	Completed	Activity	Description	Page
_____	_____	8-D	Programmed Text: Floating Add Troubleshooting. This activity allows you to develop some basic methods of troubleshooting the Floating Add Functional Unit.	8-55

LEARNING ACTIVITY 8-A. PROGRAMMED TEXT: FLOATING ADD INSTRUCTIONS

This activity reviews the function of the Floating Add instructions. You also examine and work at least one example.

OBJECTIVE

- You will be able to translate and interpret the CPU 30 through 35 instructions.

FLOATING ADD INSTRUCTIONS

The following are the six instructions executed by the Floating Add Functional Unit.

- 30 - Floating Sum
- 31 - Floating Difference
- 32 - Floating Double Precision Sum
- 33 - Floating Double Precision Difference
- 34 - Round Floating Sum
- 35 - Round Floating Difference

The 30 instruction, for example, causes the Floating Add Unit to read two operands from the Xj and Xk Registers. These operands are in floating point format and are not necessarily normalized. The sum or results of Xj and Xk are delivered to the Xi Register in floating point format and again are not necessarily normalized. The two operands are unpacked/unbiased from their floating point formats and the exponents are compared to see which is the largest and the difference is obtained.

The coefficient with the smallest exponent is right shifted by the amount of difference between the two exponents. This shift makes both coefficients' exponents equal to the same value. The two coefficients are added and form a 96-bit result. The upper half, bits 48 through 95, is selected as the coefficient and packed/ rebiased with the larger of the two exponents to form the result and sent to Xi. If Coefficient Overflow occurs, the result is right-shifted one place and the exponents value is increased by a value of one.

If the two operands are normalized but have unlike signs, the resulting coefficient may have leading zeros. No normalizing operation is built into the instruction or unit to correct this situation. You must use a separate normalized instruction in your program if the result is to be kept in a normalized form.

When the difference between the two exponents is greater than 128_{10} the unit enters a shifted operand of plus zero regardless of the sign of the shifted operand. If the reference operand (coefficient with the larger exponent) has a zero coefficient, the results can differ in sign.

Infinite (3777 XXXX....X or 4000 XXXX....X) operands or Indefinite (1777 XXXX....X or 6000 XXXX....X) operands cause bits to be set in the PSD Register for the corresponding conditions.

The following is a review of single and double precision, Coefficient Overflow, and End Around Carry. Figure 8-1 shows the 96-bit Coefficient Result Format.

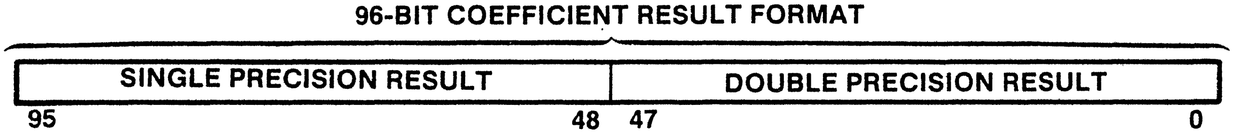


Figure 8-1. Single and Double Precision Result

Before looking at rules and examples, take a look at part of the big adder registers. For the purpose of computing floating point problems it will be 99 bits long. It will resemble figure 8-2 with the exception of only sign bits as shown below.

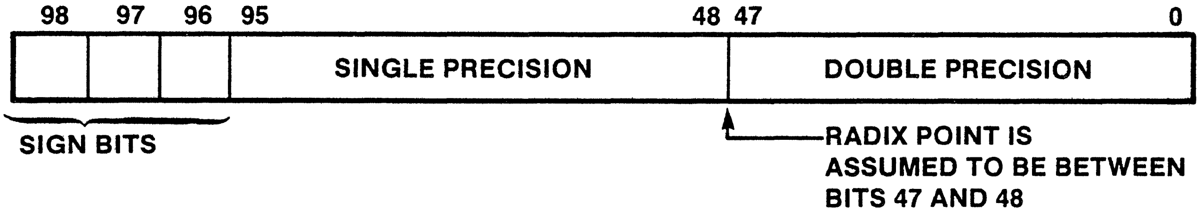


Figure 8-2. Example of Large Adder Register

In the floating add sequence, the hardware loads the operand bits 0 through 47 into the upper part of the adder registers. Bits 48 through 95, and sign bits go into bit positions 0 through 47 and 96 through 98. During this sequence the End Around Borrow (EAB) and Coefficient Overflow must be treated.

Figure 8-6 explains both of these cases. Look at Overflow first and item a(1) in figure 8-6. This shows the 3 upper sign bits as they were in figure 8-2. In this case it shows that both operands have positive coefficients so the sign bits are equal to zeros. Also there is no Borrow In from Bit 95 ($\overline{B}$) so there will be no Overflow ($\overline{\text{OVERFLOW}}$). Looking at the example in figure 8-3, assume the following operands.

98	97	96	95	94	48	Λ	47	0
0	0	0	3	0	0	Λ	0	0
0	0	0	3	0	0	Λ	0	0
0	0	0	6	0	0	Λ	0	0

Figure 8-3. Floating Add Sequence

In this example, 3+3=6 so there was no bit carried into the next higher bit.

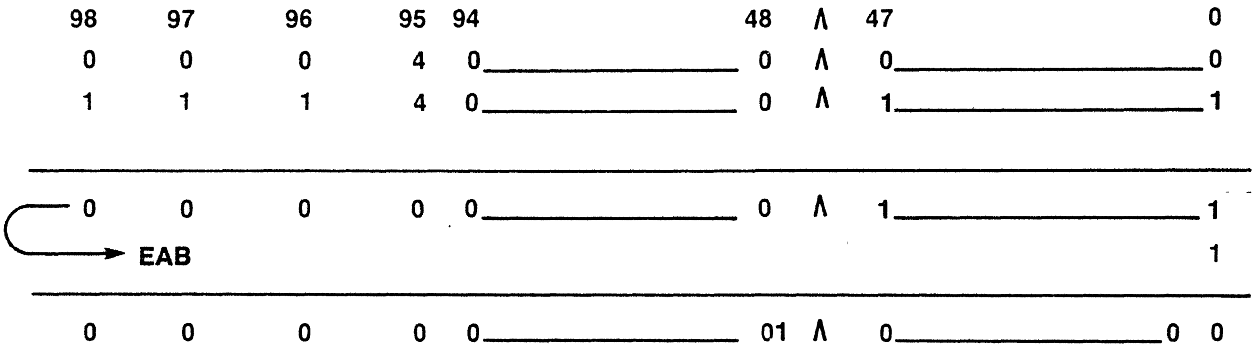
Look at item a(5) in figure 8-6. This example says if you have a Borrow from bit 95 (B) you will have Overflow (Overflow). Looking at the example in figure 8-4, assume the following operands which have positive coefficients.

98	97	96	95	94	48	Λ	47	0
0	0	0	4	0	0	Λ	0	0
0	0	0	4	0	0	Λ	0	0
0	0	1	0	0	0	Λ	0	0

Figure 8-4. Floating Add Sequence Overflow

In this example 4+4=10 so your most significant bit (MSB) of your coefficient is lost. Actually, it is in bit 96. To retrieve it, right shift one place and change the exponent by one. The value of the number has not changed because you changed the exponent. The secret of knowing when you have Overflow is when bit 96 is not equal to bit 97. So no matter what the sign bits are, you look for bit 96 not equal to 97.

A bit generated out of the coefficient satisfied in the sign bits can cause Overflow. EAB happens when that bit from the coefficient is passed all the way through the sign bits. Item b(2) in figure 8-6 shows End Around Borrow (EAB). This bit is then added to bit 0 of the noncoefficient. Item b(1) with the 3 sign bits equal to zero and a Borrow from bit 95 into bit 96 it will not have an End Around Borrow (End Around). But there will be Overflow. Look at item a(5) in figure 8-6 to see this. Look at item b(3) in figure 8-6; this shows one coefficient is positive and the other is negative and a Carry from the coefficient. Look at the following example in figure 8-5.



```
   98   97   96   95 94          48  Λ  47          0
    0    0    0    4  0________  0  Λ  0__________0
    1    1    1    4  0________  0  Λ  1__________1
 ____________________________________________________
    0    0    0    0  0________  0  Λ  1__________1
   EAB                                            1
 ____________________________________________________
    0    0    0    0  0________ 01  Λ  0________0 0
```

Figure 8-5. Floating Add Sequence EAB

In this case 4+4=10 and this bit was carried into the sign bits. It was also passed through so it became an EAB. It was then added to bit 0 of the noncoefficient.

a. OVERFLOW

B=BORROW FROM 2^{95}

SIGN BITS

(1) 98 97 96
0 0 0 $\overline{\text{B}}=\overline{\text{OVERFLOW}}$
0 0 0

(2) 98 97 96
1 1 1 $\overline{\text{B}}$=OVERFLOW
1 1 1

(3) 98 97 96
0 0 0 $\overline{\text{B}}=\overline{\text{OVERFLOW}}$
1 1 1

(4) 98 97 96
1 1 1 $\overline{\text{B}}=\overline{\text{OVERFLOW}}$
0 0 0

SIGN BITS

(5) 98 97 96
0 0 0 B=OVERFLOW
0 0 0

(6) 98 97 96
1 1 1 B=$\overline{\text{OVERFLOW}}$
1 1 1

(7) 98 97 96
0 0 0 B=$\overline{\text{OVERFLOW}}$
1 1 1

(8) 98 97 96
1 1 1 B=$\overline{\text{OVERFLOW}}$
0 0 0

b. END AROUND BORROW (EAB)

B=BORROW FROM 2^{95}

SIGN BITS

(1) 98 97 96
0 0 0 B=$\overline{\text{END AROUND}}$
0 0 0

(2) 98 97 96
1 1 1 B=END AROUND
1 1 1

(3) 98 97 96
0 0 0 B=END AROUND
1 1 1

(4) 98 97 96
1 1 1 B=END AROUND
0 0 0

SIGN BITS

(5) 98 97 96
0 0 0 B=$\overline{\text{END AROUND}}$
0 0 0

(6) 98 97 96
1 1 1 $\overline{\text{B}}$=END AROUND
1 1 1

(7) 98 97 96
0 0 0 $\overline{\text{B}}=\overline{\text{END AROUND}}$
1 1 1

(8) 98 97 96
1 1 1 $\overline{\text{B}}=\overline{\text{END AROUND}}$
0 0 0

Figure 8-6. Overflow And End Around Rules

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

Directions: Read Special Case operands in section 5 of the CYBER 170 (Model 720, 730, 750, and 760) Model 176, Hardware Reference Manual, publication number 60456100. For more information, on the Add instruction, read section 4.

The following are the basic rules for floating add numbers.

1. Determine the larger operand by unpacking to find the magnitude of each exponent.

2. Determine the difference between the exponent magnitude. This is the Shift Count for the smaller value coefficient.

3. Place the larger operand into the upper portion of the adder with three sign bits above the 48 bit operand, and 48 sign bits below or bits 0 through 47.

4. Right shift the smaller operand the difference in exponent magnitude, placing necessary sign bits above and below the operand.

5. If Coefficient Overflow occurs (that is, bit 96 is not equal to 97). Refer to figure 8-6:

   a. Right shift the sum one place.

   b. Add one to the larger exponent.

   c. Bit 98 of result denotes the sign of coefficient.

6. If the two operands are of equal magnitude but have opposite signs, the resulting sum will have a zero coefficient.

7. If the exponents of both operands are equal and no coefficient overflow occurs, it is an integer addition of the coefficient.

8. If the exponents are equal with unlike coefficient signs, take the difference of the two coefficients and attach the sign of the larger.

RULES: FLOATING ADD 30 INSTRUCTION

The following are two rule by rule examples to illustrate the Floating Add 30 instruction.

```
Example  1.   Xj  =  1717  5431  1111  1111  1111
              Xk  =  1723  4311  1111  1111  1111

Rules 1 and 2:   Xj  =  1777        Xk  =  1777        Xj  =   0060
                       -1717              -1723        Xk  =  -0054
                       -0060              -0054     Diff. =   0004

Rule 3:  Xk is larger = 000¦4311 1111 1111 1111∧0000 0000 0000 0000
                           ¦
Rule 4:  Xj RS four        ¦
                           ¦
         places =       000¦0261 4444 4444 4444∧4400 0000 0000 0000
                           ¦
                        ---------------------------------------------
Rule 5:  No overflow    000¦4572 5555 5555 5555∧4400 0000 0000 0000

Xi  =  1723  4572  5555  5555  5555


Example  2.   Xj  =  2000  6000  0000  0000  0000
              Xk  =  1776  7777  7777  7777  7777

Rules 1 and 2:      Xj  =  2000        Xk  =  1777       Xj  =   0000
                          -2000              -1776       Xk  =  -0001
                           0000              -0001    Diff.  =   0001

Rule 3:  Xj is larger  = 000¦6000 0000 0000 0000∧0000 0000 0000 0000
                            ¦
Rule 4:  Xk RS one          ¦
         place =         000¦3777 7777 7777 7777∧4000 0000 0000 0000
                            ¦
                         ---------------------------------------------
Rule 5:  Overflow        001¦1777 7777 7777 7777∧4000 0000 0000 0000
```

Note the upper three sign bits.
Bit 96 does not equal 97 so there
is Overflow. Bit 98 is a zero
denoting a positive coefficient.

```
Rule 5 RS sum one place = 000¦4777 7777 7777 7777∧6000 0000 0000 0000

Rule 5  Add one to the larger exponent  =  0000
                                          +   1
                                           0001

             Repack exponent               2000
                                          +0001
                                           2001

         Xi  =  2001  4777  7777  7777  7777
```

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

EXERCISE: FLOATING ADD 30 INSTRUCTIONS

This exercise tests your understanding of the Floating Add operation.

Directions: Work the following problems. Check your answers with those given at the end of this exercise.

1. Xj = 1717 5431 1111 1111 1111
   Xk = 1723 4311 1111 1111 1111

```
1777     1777     60
1717     1723     54
----     ----     --
  60       54      4
```

```
          5431 1111 1111 1111
R5.4
          0261 4444 4444 4444 4400
     Xi = 43 1111 1111 1111 1111 0000
     1717 4572 5555 5555 5555 ~~44~~00
```

2. Xj = 2000 4000 0000 0000 0000
   Xk = 5771 3777 7777 7777 7777

2006
2000
———
6

$\boxed{7}$ 13777 7777 7777 7777,
$\boxed{0}$ 0040 0000 0000 0000
———
7 14037 7777 7777 7777,

5771 4037 ——→ 7

Xi =

3. Xj = 2000 6000 0000 0000 0000
   Xk = 1776 7777 7777 7777 7777

2000   1777
2000   1776
———   ———
0   — 1

2000
+1
———
2001

$\boxed{}$ 6000
1 3777 — 4
———
$\boxed{1}$ 1777 — 4

Xi =
RS=1

If you have any questions, ask your technical advisor.

2001 4777 7777 7777 7777, 6000

ANSWERS FOR FLOATING ADD EXERCISE

1. Xj has positive coefficient and negative exponent; Xk has the same.

```
Xj exponent = 1717 so  1777
                      -1717
                      -  60
Xk exponent = 1723 so  1777
                      -1723
                      -  54
```

In this case Xj is the smallest.

```
Xj =  60
Xk = -54
       4
```

There is a difference of four so right shift Xj four places.

```
Xk = 000¦4311 1111 1111 1111∧0000 0000 0000 0000
Xj = 000¦0261 4444 4444 4444∧4400 0000 0000 0000
     000¦4572 5555 5555 5555∧4400 0000 0000 0000
```

There was no EAB or Overflow.

Xi = 1723 4572 5555 5555 5555

2. Xj has a positive coefficient and positive exponent. Xk has a negative coefficient and positive exponent.

```
Xj exponent = 2000
             -2000                        In this case Xj is
              0000                        the smallest and
Xk exponent = 5771 complement to 2006     shift Xj right six
                                -2000     places.
                                +   6
```

Since Xk is negative its sign bits will be all ones.

```
Xk = 111¦3777 7777 7777 7777∧7777 7777 7777 7777
Xj = 000¦0040 0000 0000 0000∧0000 0000 0000 0000
     111¦4037 7777 7777 7777∧7777 7777 7777 7777
```

There is no EAB or Overflow but upper sign bit is equal to a one so coefficient is negative.

Xi = 5771 4037 7777 7777 7777

ANSWERS FOR FLOATING ADD EXERCISE (Contd)

3. Xj has a positive coefficient and positive exponent.
Xk has a positive coefficient and negative exponent.

```
Xj Exponent = 2000
             -2000
             -----
              0000                Xj is the largest and
Xk Exponent = 1776 so  1777       shift Xk by one.
                      -1776
                      -----
                      -0001

000¦6000 0000 0000 0000∧0000 0000 0000 0000
000¦3777 7777 7777 7777∧4000 0000 0000 0000
-------------------------------------------
001¦1777 7777 7777 7777∧4000 0000 0000 0000
```

There is no EAB but you do have Overflow since bit 96 is not equal to 97 so you must RS one and add one to the exponent.

Sum equals

```
000 4777 7777 7777 7777  6000 0000 0000 0000

Xi = 2001 4777 7777 7777 7777
```

RULES: FLOATING DIFFERENCE 31 INSTRUCTION

The following are the basic rules for Floating Difference 31 Instruction and an example to illustrate the operation.

1. Complement the Xk-operand coefficient, including sign bits above and below; proceed as in the sum operation.

2. Alignment and Overflow operations are the same as Floating Sum. Rules 1 through 5.

3. If the two operands are identical, the result will be a positive coefficient of zero with the same exponent.

4. If the exponents are both zero (2000) and no Overflow occurs, it is an ordinary integer subtraction.

The following are examples.

o Xj = 1717 5431 1111 1111 1111
Xk = 1723 4311 1111 1111 1111

```
Xj  =   1777   Xk =   1777        Xj   =   0060
       -1717         -1723        Xk   =  -0054
       -----         -----                -----
       -0060         -0054        Diff. =  0004
```

Complement

```
Xk = 111¦3466 6666 6666 6666∧7777 7777 7777 7777
Xj = 000¦0261 4444 4444 4444∧4400 0000 0000 0000
     ---------------------------------------------
     111¦3750 3333 3333 3333∧4377 7777 7777 7777
```

Note the 3 upper sign bits.

Bit 96 = 97 so there is no Overflow.

Bit 98 is a 1, which denotes a negative coefficient.

Xi = 6054 3750 3333 3333 3333

EXERCISE: FLOATING DIFFERENCE 31 INSTRUCTION

This exercise tests your understanding of the Floating Difference operation.

Directions: Work the following problems. Check your answers with those given at the end of this exercise.

1. Xj = 1721 6000 0000 0000 0000
   Xk = 1723 4000 0000 0000 0000

1777 1777
1721 1723
-56 -54 2

111 | 3777 ?A7
000 | 140 ?A6
111 5377 ?A7

Xi =

1723
(6

6054 537 7,7777 7777 7777

2. Xj = 6050 2777 7777 7777 7777
   Xk = 1721 6000 0000 0000 0000

6050 1777 1777
6 1727 1721
1727 —50 → 56

XK 6 0060

111 | 7717 7777 → ∧7
Xi= 111 | 3777 7 777 → ∧77
111 2717 7 76

6054 2717 7777 7777 7777 7777 1

3. Xj = 2006 1000 0000 0000 0777
   Xk = 2022 1000 0000 0000 0777

2006 2022
2000 2000
6 22

XK 111 | 6777 7777 7777 7000∧77
000 | 0 0000 1000 0000 0000∧17
111 | 7000 0777 7777 700ϕ∧7

2022
6
5755 7000 0777 7777 7001

Xi =

If you have any questions, see your technical advisor.

The answers for this exercise are on the following page.

ANSWERS FOR FLOATING DIFFERENCE 31 INSTRUCTIONS EXERCISE

1. Xj has a positive coefficient and negative exponent; Xk has the same.

```
Xj  =  1777  Xk  =  1777
      -1721        -1723
      -  56        -  54
```

Xk greater so shift Xj two places right

Complement

```
Xk = 111 ¦ 3777  7777  7777  7777 ∧ 7777  7777  7777  7777
     000 ¦ 1400  0000  0000  0000 ∧ 0000  0000  0000  0000
     111 ¦ 5377  7777  7777  7777 ∧ 7777  7777  7777  7777
```

There is no Overflow or EAB but the coefficient is now negative as bit 98 is a one. The exponent 1723 is complemented to 6054.

```
Xi = 6054  5377  7777  7777  7777
```

2. Xj has negative coefficient and negative exponent. Xk has positive coefficient and negative exponent.

```
Xj  =  6050 complement to 1727   1777  Xk  =  1721   1777
                                -1727               -1721
                                -  50               -  56
```

Xj = Greater shift Xk six places right

Complement

```
Xj  111 ¦ 2777  7777  7777  7777 ∧ 7777  7777  7777  7777
Xk  111 ¦ 7717  7777  7777  7777 ∧ 7777  7777  7777  7777
    111 ¦ 2717  7777  7777  7777 ∧ 7777  7777  7777  7776
        ¦ EAB                                            1
    111 ¦ 2717  7777  7777  7777 ∧ 7777  7777  7777  7777
```

There was EAB but no Overflow.

```
Xi = 6050  2717  7777  7777  7777
```

ANSWERS FOR FLOATING DIFFERENCE 31 INSTRUCTIONS FOR EXERCISE (Contd)

3. Xj has a positive coefficient and positive exponent; Xk has the same.

```
Xj  =  2006  Xk  =  2022     Xk  Greater so shift
      -2000        -2000     Xj  14 places
      +   6        +  22
```

```
Xk   111 ¦ 6777  7777  7777  7000 ∧ 7777  7777  7777  7777
Xj   000 ¦ 0000  1000  0000  0000 ∧ 0777  0000  0000  0000
     111 ¦ 7000  0777  7777  7001 ∧ 0776  7777  7777  7777
```

There is no Overflow or EAB but the coefficient is now negative as bit 98 is a one. The exponent 2022 is complemented to 5755.

```
Xi  =  5755  7000  0777  7777  7001
```

RULES: FLOATING DOUBLE PRECISION ADD 32 INSTRUCTION

The following are the basic rules for Floating Double Precision Add and an example to illustrate the operation.

1. This instruction forms the sum of two floating point numbers as in the floating sum instruction, but packs the lower half of the double precision sum with an exponent 60_8 less than that for the upper sum.

2. Special situations are the same as for single precision.

3. If one of the operands is at the upper limits of the floating point range, single precision results may be out of range; but since the double precision results are minus 60_8, no Overflow is generated.

4. If the two operands are near the lower limits, then double precision can cause Underflow. If the result is -1777_8 unbiased, the result is processed as a normal floating point operation. The sign of the result is the same as the sign of the larger exponent. If the exponents have identical value but opposite signs, the result has the sign of Xk.

Example:

```
1.        Xj  =  1646  7777  7777  7777  7777
          Xk  =  1567  7777  7777  7777  7777

Xj   =   1777  Xk   =   1777   Xk   =   0210
        -1646          -1567   Xj   =  -0131
        -0131          -0210 Diff.  =   0057

Xj is larger =  000 : 7777  7777  7777  7777 ^ 0000  0000  0000  0000
XK  RS   57  =  000 : 0000  0000  0000  0001 ^ 7777  7777  7777  7776
Places              :
Have Overflow   001 : 0000  0000  0000  0000 ^ 7777  7777  7777  7776
96=97               :
RS sum one          :
  place         000 : 4000  0000  0000  0000 ^ 3777  7777  7777  7777

Xj   =  -  131                                  To find new exponent
        -   60  Because of double precision           1777
        -  211                                    -    210
        +    1  Because of Overflow                   1567
        -  210

        Xi   =  1567  3777  7777  7777  7777
```

EXERCISE: FLOATING DOUBLE PRECISION ADD 32 INSTRUCTION

This exercise tests your understanding of the Floating Double Precision Add.

Direction: Work the following problem. Check the answer with that given on the following page.

```
    1.   Xj  =  1717  5431  1111  1111  1111
         Xk  =  1723  4311  1111  1111  1111
```

```
 1777      1777
 1717      1723
 ----      ----
- 60      - 54          4

     .54

Xk 000|4 311 1111 1111 1111 A0 000
   000 0 261 4444 4444 4444 44 00
       4 572 7555 7555 5556 44
```

```
 1723
→  60
 ----
 1643
```

Xi =

1643 4400 0000 0000 0000

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

ANSWER FOR FLOATING DOUBLE PRECISION ADD EXERCISE

1. Xj has positive coefficient and negative exponent and Xk has the same.

   Xj Exponent = 1717 so

```
 1777
-1717
-  60
```

```
Xk Exponent  =  1723 so  1777
                        -1723
                        -  54
```

In this case Xj is the smallest.

```
Xj  =     60
Xk  =  -  54
           4
```

The difference of four, so right shift Xj four places.

```
Xk  =  000 ¦ 4311  1111  1111  1111 ^ 0000  0000  0000  0000
Xj  =  000 ¦ 0261  4444  4444  4444 ^ 4400  0000  0000  0000
       000 ¦ 4572  5555  5555  5555 ^ 4400  0000  0000  0000
```

There was no EAB or Overflow.

```
Xk  =  -  54
       -  60   Because of double precision
       - 134
```

To find new exponent

```
 1777
- 134
 1643
```

Xi = 1643 4400 0000 0000 0000

RULES: FLOATING DOUBLE PRECISION DIFFERENCE 33 INSTRUCTION

The following are the basic rules for Floating Double Precision Difference and an example to illustrate the operation.

1. The instruction forms the difference between two operands as in the 31 instruction, but packs the lower half of the Double Precision Difference with an exponent of 60_8 less than the upper difference.

2. If the two operands are identical, the coefficient result will be zero, the exponent will be the same as for a non-zero coefficient. The sign of the coefficient will be zero.

Example:

```
1.   Xj  =  6056  3677  7777  7777  1234
     Xk  =  5761  3421  7777  7777  4016

     Xj  =  1777          Xk  =  2016              - 0056
           -1721                -2000              + 0016
           -0056                +0016  Diff.       = 0074

Xk complemented=000 ¦ 4356  0000  0000  3761 ∧ 0000  0000  0000  0000
Xj RS 74 places=111 ¦ 7777  7777  7777  7777 ∧ 7777  3677  7777  7777
                000 ¦ 4356  0000  0000  3760 ∧ 7777  3677  7777  7777
                  └─¦─> EAB                                         1
                000 ¦ 4356  0000  0000  3760 ∧ 7777  3700  0000  0000

     Xk  =  + 0016
            - 0060   Because of double precision
            - 0042

     To find new exponent   1777
                           -0042
                            1735

     Xi  =  1735  7777  3700  0000  0000
```

111 | 3421
000 | 4356

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

EXERCISE: FLOATING DOUBLE PRECISION DIFFERENCE 33 INSTRUCTION

This exercise tests your understanding of the Floating Double Precision Difference.

Directions: Work the following problem. Check your answer with that given at the end of this exercise.

```
     1.   Xj  =  1721  6000  0000  0000  0000
          Xk  =  1723  4000  0000  0000  0000
```

```
  1777        1777
  1721        1723
-   56      -   54

111 | 3 777 7777 7777 ∧77 -
    | 1 000 0000 0000  000
①11   4 7777 7777 7777 777
```

Xi =

```
-54       1777
-60      -134
-134      1643
         6
         6134 7777 7777 7777 7777
```

The answer for this exercise is on the following page.

ANSWER FOR FLOATING DOUBLE PRECISION DIFFERENCE EXERCISE

1. Xj has a positive coefficient and a negative exponent; Xk has the same.

```
Xj Exponent  =  1721 so  1777
                        -1721
                        -  56

Xk Exponent  =  1723 so  1777
                        -1723
                        -  54
```

In this case Xj is the smallest.

```
Xj  =    56
Xk  =  - 54
          2
```

The difference of two, so right shift Xj two places.

Complement Xk =

```
111 ¦ 3777  7777  7777  7777 ∧ 7777  7777  7777  7777
000 ¦ 1400  0000  0000  0000 ∧ 0000  0000  0000  0000
111 ¦ 5377  7777  7777  7777 ∧ 7777  7777  7777  7777
```

There is no EAB or Overflow, but the coefficient is now negative as bit 98 is equal to a one. The exponent is complemented after double precision correction.

```
Xk  =  - 54
       - 60
       -134
```

To find new exponent

```
 1777
- 134
 1643
```

Complement to 6134

Xi = 6134 7777 7777 7777 7777

RULES: ROUND FLOATING SUM 34 INSTRUCTION

The following are the basic rules for Round Floating Sum and an example to illustrate the operation.

1. The rules for adding are the same as Floating Sum.

2. The round bit is always inserted in the coefficient with the largest exponent.

3. If the two exponents are equal the round bit is inserted in the Xk coefficient.

4. The round bit is always the complement of the coefficients sign bit and inserted to the right of the lowest-order bit in the coefficent, or bit 2^{-1}. This is done before shifting.

5. A second round bit is added to the other coefficient if both operands are normalized or have unlike signs.

Example:

```
     Xj  =  2060  4444  4444  4444  4444
     Xk  =  2063  4444  4444  4444  4444

     Xj  =  2060     Xk  =  2063     Xk is the largest and
           -2000           -2000     Xj is RS three places.
           +  60           +  63

                                        ROUND BIT
Xk is larger  =  000 ¦ 4444  4444  4444  4444 ∧ 4000  0000  0000  0000
Xj RS three          ¦
places =         000 ¦ 0444  4444  4444  4444 ∧ 4000  0000  0000  0000
                 ------------------------------------------------------
                 000 ¦ 5111  1111  1111  1111 ∧ 0000  0000  0000  0000
```

There is no EAB or Overflow.

```
Xi  =  2063  5111  1111  1111  1111
```

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

EXERCISE: ROUND FLOATING SUM 34 INSTRUCTION

This problem tests your understanding of the Round Floating Sum.

Directions: Work the following problem. Check your answer with that given at the end of this exercise.

```
     1.  Xj  =  3776  0000  0000  0000  0001
         Xk  =  3771  0000  0000  0000  0077
```

```
  3776          3771
- 2000          2000
  ----          ----
  1776          1771

                5
```

Xi=

```
0000 0000 0000 0001∧4 0 0
0000 0000 0000 0001A7 6 0
---------------------------
                 03A3 6 0

3776 0000 0000 00003
```

The answer for this learning activity is on the following page.

ANSWER FOR ROUND FLOATING SUM EXERCISE

1. Both Xj and Xk have positive coefficients and exponents.

Xj Exponent = 3776

```
so  3776
   -2000
   +1776
```

Xk Exponent = 3771

```
so  3771
   -2000
   +1771
```

In this case Xj is the largest and gets the round bit.

```
Xj  =  1776
Xk  = -1771
          5    The difference is five so right shift
               Xk five places.
```

```
Xj  =  000 ¦ 0000  0000  0000  0001 ∧ 4000  0000  0000  0000
Xk  =  000 ¦ 0000  0000  0000  0001 ∧ 7600  0000  0000  0000
       000 ¦ 0000  0000  0000  0003 ∧ 3600  0000  0000  0000
```

There is no EAB or Overflow.

```
Xi  =  3776  0000  0000  0000  0003
```

RULES: ROUND FLOATING DIFFERENCE 35 INSTRUCTION

The following are the basic rules for Round Floating Difference and an example to illustrate the operation.

1. The rules for subtracting are the same as Floating Difference.

2. The rounding rules are the same as Round Floating Sum with the exception of rule 5 which is as follows:

   A second round bit is added to the other coefficient if both operands are normalized or have like signs.

Example:

```
Xj  =  1723  4000  0000  0000  4444
Xk  =  1720  4000  0000  0000  7777

Xj  =  1723 so  1777
               -1723
               -  54

Xk  = 1720  =  1777
            . -1720
              -  57
```

Xj is the largest and Xk is RS three places.

Since the signs of the coefficients are like signs you have two round bits.

```
Xj is larger     000 ¦ 4000  0000  0000  4444 ∧ 4000  0000  0000  0000
Xk RS three          ¦
 Places          111 ¦ 7377  7777  7777  7000 ∧ 0377  7777  7777  7777
And Complemented     ¦
                 000 ¦ 3400  0000  0000  3444 ∧ 4377  7777  7777  7777
                  ───→EAB                                            1
                 000 ¦ 3400  0000  0000  3444 ∧ 4400  0000  0000  0000
```

There is no Overflow.

```
Xi  =  1723  3400  0000  0000  3444
```

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-A

EXERCISE: ROUND FLOATING DIFFERENCE 35 INSTRUCTION

This problem tests your understanding of the Round Floating Difference.

Directions: Work the following problem. Check your answer with that given at the end of this exercise.

```
1. Xj  =  1723  3777  7777  7777  7777^4
   Xk  =  1726  3400  7777  7777  7776^4
```

```
 1777        1777
 1723        1726
 ----        ----
 - 54        - 51
```

~~R5=7 0 0377777 7777 7777^7 400~~

Xi=

~~111|7400 0000 0000 0000 0377~~
~~Xj |3400 7777 7777 7776^4000 -~~
~~000 3000 7777 7777 7776 4377~~
~~1~~
~~3000 7777 7777 7776^4400~~

~~1726 3000 7777 7777 7776~~

```
XJ 000|03 777777  7777  7777^740
   111|43 770000  0000  0001 3777
   ------------------------------
   (1)11 47770000 0000  0001^3337
```

1726
76
6051 4777 0000 0000 0001

The answer for this learning activity is on the following page.

ANSWER FOR ROUND FLOATING DIFFERENCE EXERCISE

1. Both Xj and Xk have positive coefficients and negative exponents.

   Xj Exponent = 1723 so $\begin{array}{r} 1777 \\ \underline{-1723} \\ -\ \ 54 \end{array}$

   Xk Exponent = 1726 so $\begin{array}{r} 1777 \\ \underline{-1726} \\ -\ \ 51 \end{array}$

   In this case Xk is the largest and coefficients signs are alike so you have two round bits.

   $$\begin{array}{lcr} Xj & = & 54 \\ Xk & = & \underline{-51} \\ & & 3 \end{array}$$

   The difference of three so right shift Xj three places.

   $$\begin{array}{lcc:ccccccccc} Xk & = & 111 & 4377 & 0000 & 0000 & 0001 & \wedge\ 3777 & 7777 & 7777 & 7777 \\ Xj\ RS3 & = & 000 & 0377 & 7777 & 7777 & 7777 & \wedge\ 7400 & 0000 & 0000 & 0000 \\ \cline{3-11} & & 111 & 4777 & 0000 & 0000 & 0001 & \wedge\ 3777 & 7777 & 7777 & 7777 \end{array}$$

   There is no EAB or Overflow. Note that bit 98 is equal to a one so the coefficient is now negative. The exponent is complemented to compensate for this.

   Xi = 6051 4777 0000 0000 0001

You have completed learning activity 8-A. You may wish to review this material or continue with the next learning activity.

LEARNING ACTIVITY 8-B. TEXT READING: FLOATING ADD INSTRUCTIONS

This activity follows the data flow and control signals used by the 30 through 35 instructions through the Floating Add Functional Unit primary block 1.0 diagram.

OBJECTIVE

- You will be able to follow the data flow and control signals through the Floating Add Unit primary block 1.0 diagram.

Directions: Place microfiche GF60420300W number 55 (4 or 10) in your microfiche viewer and select coordinate B3/B4.

The Floating Add instruction requires four clock periods to complete execution. The 30 and 34 instructions are identical except the 34 instruction adds the round bit to the coefficient. The 30 and 32 instructions are also identical except the 32 instruction takes the results from the lower 48 bits of the 96-bit register. The same holds true for the Difference instructions. The 31 instruction is different only from the 35 instruction because the 35 instruction rounds. The 33 instruction is different from the 31 instruction because the 33 instruction takes the lower 48 bits for the result. When doing a Difference instruction the Xk operand is complemented.

DIAGRAM LAYOUT

This diagram is laid out in the conventional way, with inputs on the left and outputs on the right. The registers and Flip-Flops are arranged in vertical columns according to their clock periods.

There are three columns of registers. The first column starting on the left in quadrant A has the ml Input F/F at the top and Borrow Register at the bottom of quadrant C. The second column in quadrant B has the Double Precision F/F at the top and Borrow Register at the bottom in quadrant D. Therefore, this diagram shows everything that happens in each clock period. Everything that happens to the left of the first column of registers including the setting of these registers takes place in the first clock period. Everything between the first and second column of registers including the setting of these registers happens in the second clock period. The third clock period involves what is between the second and third column of registers including the setting of the result register in the CPU, which takes place during the fourth and last clock period. The exponent takes a path in the lower portion of the upper half of the diagram.

The coefficients path is in the lower half. Shift count determination and shift control is across the middle. The rest of control is across the top of the diagram.

INPUT REGISTERS

Starting at the top of quadrant A are the m1 and m2 Input Flip Flops. These bits are from the CIW Register in the CPU. Next are the Xk Input and Xj Input Registers. These receive both operands from the X Registers in the CPU. Last are two Enable and Borrow Registers that hold the partial difference of the Xj and Xk operand exponents.

In quadrant B is GO FLOATING ADD F/F, which sets when receiving GO FLTG ADD from the CIW Register in the CPU.

CONTROL

Since the number of controls are increasing, they will be covered and used as you progress through the diagrams. Many of them are similar to those used in the Normalize Unit.

METHOD

In quadrant A the two 60-bit operands from Xj and Xk are received during the first clock period. If m0 is equal to a one, this would mean a 31, 33, or 35 instruction, which would be a Difference. This would cause Xk to be complemented. During the second clock period these two coefficients go to the Select Operand logic in quadrant C. Also during the first clock period m1 and m2 are used in controlling the six different instructions. In quadrant C bits 48 through 59 enter the unit from the Xj and Xk Registers in the CPU. Bits 48 through 58 are packed or biased exponents. These values are complemented if bit 59 equals a one, meaning the coefficient is negative. Xk bits are always complemented so the adder subtracts. The adders output is used in Shift Ranks 1, 2, and 3. It also outputs a signal called SIGN OF DIFFERENCE. When this signal is equal to a one Xj is the larger exponent.

A second stage for subtracting the exponents' values is in quadrants A and C. Its output is used in Shift Ranks 4 and 5. The Shift 128 blocks the output Shift Rank 5 causing the shifted operand to become a positive zero if the difference between exponents is greater than 128_8. The output of the shift ranks is complemented and is inputted to the adder. This shifted operand is now equal in value to the reference operand. The reference operand leaves the Select Operand logic to the Reference Register in quadrant D. During the third clock period it is added in the Shifted Operand plus Reference Operand first stage adder. On the fourth clock period the coefficient is completely added and sent to the Xi Register in the CPU.

In a single-precision operation bits 48 through 95 are selected. In a double-precision operation bits 0 through 47 are selected. Above the coefficient adder is a Borrow to 96 Propagation Network that outputs to Coefficient Overflow Test. With Coefficient Overflow, the most significant bit is lost. The examples shown were right shifted one place. In reality the hardware takes bits 48 through 96 if single precision and bits 1 through 48 if double precision.

In quadrant A and the Xk and Xj Input Registers the coefficient moves down to the Select Operand logic in quadrant C, and the exponent continues to the right, where the bias is removed. The two exponents go to the Select Exponent logic. This contains the SIGN OF DIFF signal and selects the exponent that is the largest. Taking bits 0 through 10, the unbiased exponent, it goes to the Selected Exponent Register. On the third clock period this exponent is complemented so it can be used as the subtrahend. If this is a single precision operation the other input is zero; but if the operation is a double-precision, it will be 60, making the exponent 60_8 smaller. This exponent is stored in the Exponent Register until the fourth clock period. In the exponent Output Control logic the exponent is biased, and the sign of the coefficient is added then sent to the Xi Register in the CPU.

In quadrant A is the m2 Input Flip Flop. If m2 equals a one, this means a rounding operation. The round bit goes to the largest or reference operand, or Round Reference to Round Bit Control logic. Normalized Operands or the Xk Sign ≠ Xj Sign Test controls the second round bit to the shifted operand if both operands are normalized, or if they are adding and the exponents have unlike signs, or if subtracting and the exponents have like signs.

The last logic in quadrant A is Exponent Indefinite, Overflow Test logic, which is checking for bad exponents. If bad, these signals go to Exponent Special Case Test logic in quadrant B. At the top of the test logic is DP or double-precision if ml equals one. The next is GO, generated by GO FLOATING ADD from the CIW in the CPU. These signals and GO ADD SPECIAL CASE go to the Coefficient Output Selection Control that determines the bits transmitted from Coefficient Output Selection in quadrant D. The other inputs, Xj and Xk Overflow and Indefinite, cause special cases. They are sent to X Sign Control in the CPU. X Sign Control decides which of the special case exponents will be placed in the Xi Register.

At the bottom of the Exponent Special Case Test logic there is an input called Exponent 2060. When doing double-precision, 60_8 is subtracted from the exponent. If, for example, the exponent unbiased was -1777 and 60_8 is subtracted, it is more negative than it can handle and therefore causes Underflow. The logic looks at any exponent more negative than 1777 or 2000.

You have completed learning activity 8-B. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 8-C. TEXT READING: FLOATING ADD DETAILED PAK DIAGRAMS

This activity follows the data flow and control signals used by the 30 through 35 instructions through the detailed pak diagrams Floating Add Functional Unit detailed pak 3.0 diagrams.

OBJECTIVE

- You will be able to follow the data flow and control signals through the Floating Add Unit detailed 3.0 diagrams.

Directions: Place microfiche GF60420300W number 55 (4 of 10) in your microfiche viewer and select coordinate C1/C2.

INPUT REGISTERS

In quadrant A and C are six FF modules at 5B02 through 5B07. Each module holds every sixth bit 0, 6, 12 and so forth. These modules receive the Xj and Xk operand during the first and same clock period that the instruction is issued from the CIW Register. Also entering are m0 and m2. When m0 is set it denotes a Difference operation and complements the Xk operand. Signal m2 goes to the m2 Input Flip Flop and is used during a round operation. Each operand is split into its own coefficient and exponent. These go to their own input registers, such as Xk Coefficient Input Register. In quadrant C is an FA module at 5B01. This module has the first stage of an adder. It finds the difference between the two exponents. The partial results are held in the Group and Bit Enable Registers and also the Bit and Group Borrow Register.

In quadrant D on the FD module at 5C04 is GO FLOATING ADD and m1. If this is a double-precision operation, m1 sets and goes to a Two Clock Holding Flip Flop where it is used during the fourth and final clock period. This two clock delay is needed as it enters the Floating Add Unit during the first clock period and is held in the GO HOLDING Flip Flop. These are all the inputs from the CPU.

CONTROL

At this time you will look only at the controls necessary for a Floating Add 30 instruction, and this will be covered as the operation progresses through the logic. At the end of the learning activity the others will be covered.

METHOD

Let's divide this operation into the four clock periods required to execute the instruction. Also let's use the first example given for a 30 instruction on page 8-9 of learning activity 8-A. The operands were as follows:

Xj = 1717 5431 1111 1111 1111
Xk = 1723 4311 1111 1111 1111

In this example the actual values of the exponents were:

Xj = -60
Xk = -54

This made Xj operand the smallest and causes the Xj coefficient to be right shifted four places.

First Clock Period: Exponent. In quadrant C on the FA module at 5B01 take Xj first whose value is 1717. Bit 58 is complemented to unbias its exponent giving a value of 3717. With Xj's coefficient positive, bit 59 is zero and bits 48 through 58 will not be complemented, so the value remains 3717. Xk value is 1723. Bit 58 is not complemented but the remaining bits 48 through 57 with a value of 0054 are. This not only removes the bias but makes it the subtrahend so the difference of the exponents can be determined. Xk value is not complemented as it goes to the adder because bit 59 equals zero. Xk value is still 0054. The partial results would be held in the Group and Bit Enable Registers and Bit and Group Borrow Registers. In quadrant A are six FF modules at 5B02 through 5B07. The Xj and Xk exponents are held in the Xj and Xk Exponent Input Registers.

First Clock Period: Coefficient. In quadrant A on the FF modules at 5B02 through 5B07. The coefficients will be held in the Xk and Xj Coefficient Input Registers.

Second Clock Period: Exponent. The FB, FD and FE modules each have a second stage of the adder. The adder is completing the subtraction of the two exponents. All three of these modules receive their Enables and Borrows from the FA module, which holds the first stage. In quadrant C the FB module at 5C01 forms the results for Shift Control 1, 2, and 4. Read note 3 on the diagram. Since Xk is the larger and the shift count is equal to four there is a four at the fanout. The other output is SIGN OF DIFFERENCE and equals a one when Xj is the largest number.

The 12 exponent bits of Xj and Xk held in the Exponent Input Registers in the FF modules in quadrant A go to the two FC modules at 5C02 and 5C03. The bias is removed by complementing bit 10. Bits 0 through 10 are then used by the second stages of the adder on the FD module at 5C04 and the FE module at 5C05. In quadrant D is the FD module at 5C04. It receives exponent bits 0 through 5 and the Group Enables and Borrows from first stage of the adder. Only bits 3, 4, and 5 go to the adder. This second stage forms the Shift Control of 8, 16, and 32 and is held in the Shift 8, 16, 32 Register. This is in true form. In quadrant B is the FE module at 5C05. It receives Group Enables and Borrows from the first stage of the adder. This module receives exponent bits 6 through 10. The second stage forms the Shift Control of 64 and 128 and will be held in the Shift 64, 128 Register. On the FC module at 5C02 and before the removing of bias, bit 11 goes to the FE module at 5C05.

The same thing is happening on 5C03. This is Xj sign bit and it is sent to 5C05. Bits 6 through 10 and the sign of the coefficient are gated by SIGN OF DIFFERENCE. The SIGN OF DIFFERENCE value is a one if Xj is the largest number. In our example Xj was smallest, so this zero would be complemented and gate Xk bits 6 through 10 and Xk sign of the selected Exponent Register and Reference Sign Flip Flop.

At the Remove Bias logic on the FC module at 5C02 the sign bit determines bits of the exponent that will be complemented. If the sign bit is positive (equal to 0), bit 10 is complemented. If sign is negative (equal to 1), bits 0 through 9 are complemented. Since the sign in our example is positive, bit 10 is complemented storing a 37 in the selected Exponent Register on the FE module at 5C05 and the sign bit equals zero. Xk and Xj exponent bits 0 through 5 are on the two FC modules at 5C02 and 5C03. These bits are sent down to the FD module at 5C04 and are gated by the SIGN OF DIFFERENCE as on the FE module and stored in the selected Exponent Register. Its value at this time is 23.

Second Clock Period: Coefficient. In quadrant C the Xj and Xk coefficient bits 0 through 47 go to the FG and FO modules on the 3.1 diagrams at coordinate D1/D2. Quadrants A and C have the FO module at 5B09 and the FG modules at 5B08 and 5B10 through 5B15. Using the FO module 5B09, look at the data flow and controls needed. In quadrant C are Xj and Xk bits 0 through 12. This splits, sending Xj and Xk to the shift ranks and Xj and Xk to the Reference Operand Register. Above the incoming data bits is the signal SIGN OF DIFFERENCE. For our example it equals a zero. This zero will be complemented allowing Xj to be gated to Shift Rank 1 and Xk to the Reference Operand Register. Shift count is the uppermost signal on the FO module. Due to Xk being the largest it is in true form equal to four. Since SIGN OF DIFFERENCE equals zero it will not be complemented. In this example, bit 0 equals a one and is shifted and becomes bit 44 of the 96-bit coefficient.

This shifted value is stored in the Shifted Operand Register. For continuity let's follow the coefficient to the point of being transmitted to the Xi Register.

Third Clock Period: Coefficient. The shifted operand goes through Shift Rank 3 to the FH modules at 5C08 through 5C15. These modules hold the Shift Ranks 4 and 5. In quadrant D is SHIFTED SIGN into the Shift Rank 4. This is used to sign extend as you right shift. Read note ①. As the shifted operand leaves these FH modules it is 96 bits in length. In quadrant C the reference operand, only 49 bits in length (or bit 47 through 95), along with the shifted operand go to the FJ modules and one FI module on the 3.2 diagrams at coordinate El/E2.

Quadrant A and C shows the FJ modules at 5D01 through 5D10 and the FI module at 5D11 in quadrant A. These modules are the first stage of the adder. In quadrant C are the shifted operand bits 0 through 95 and reference operands bits 47 through 95. The Reference Sign to FJ modules 5D01 through 5D06 gives you the sign bits for bits 0 through 46. Sign bit 47, since it can be a round bit, was taken care of on the 3.1 diagrams on the FO module at 5B09. The FI module in quadrant A receives both Shifted and Reference Sign. These become bits 96 of both operands. Read note 5.

Since the upper two bits of the adder are sign bits, the adder will always have an End-Around Borrow, or EAC as in the examples, if both operands are positive. If both operands are negative, the adder will never have an End-Around. If the signs are unlike, the remaining bits control the End-Around Borrow. Both operands are complemented prior to being added. The FJ modules handle nine bits and the FI module handles only six bits. This adder is very similar to the Long Add Units adder. The first stage divides the operand into bits, groups, and sections that form a partial sum consisting of Enables and Borrows as shown in figure 8-7.

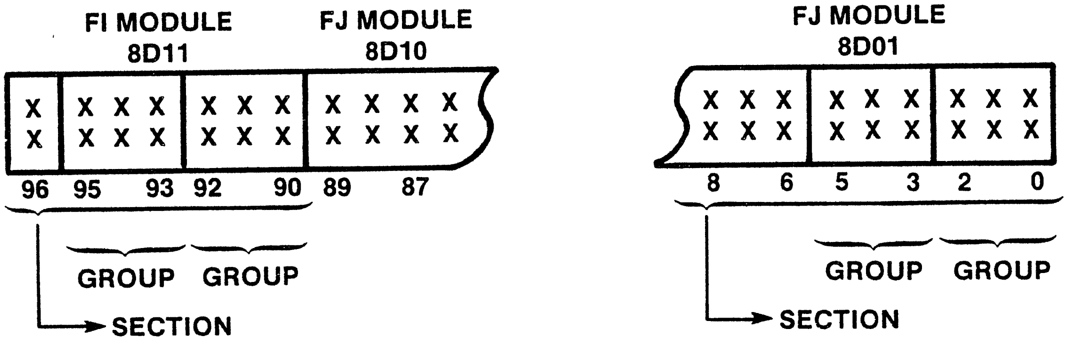


Figure 8-7. Adder Format

Each module is a section, therefore, sections 1 through 10 are nine bits long, because they are on the FJ modules. Section 11 on the FI module is 7 bits long. A Section Borrow will be set or equal to a one when a borrow out of that section occurs. A Section Enable will be set or equal to a one when all bit Enables in that section are equal to one.

A group is three bits long. There are only two groups per section (refer to figure 8-7, Adder Format). These two groups are in the lower six bits in each section. A Group Enable will be set or equal to a one when all bit Enables in that group are set equal to one. A Group Borrow will be set or equal to a one when a Borrow out of that group occurs (refer to Long Add description of adds in learning activity 2-B). This partial sum is held in the three Enable and three Borrow Registers and will be used during the fourth and final clock period.

In the upper portion of quadrant A is a FL module 5D12. It detects Coefficient Overflow. The Coefficient Overflow Detecting Network is divided into two parts. The first part tests to see if a Borrow will be propagated up to bit 96. The BORROW TO 96 signal and the signs of the two operands are sent to the second part Coefficient Overflow Detection. The translation for Overflow is described on the diagrams (refer to figure 8-6, Overflow and End Around Rules). To have coefficient Overflow, the signs of the two operands are positive or equal to zeros, the network looks for $\overline{\text{BORROW TO 96}}$. If the signs are both negative or equal to ones, the network looks for BORROW TO 96.

Fourth Clock Period: Coefficient. The partial sum leaves the FJ and FI modules and goes to the second stage on the FK modules at 5E01 through 5E11.

There are four different data paths for this 97-bit result. They are controlled by four signals on the following paths.

- Single-Precision - Uses output bits 48 through 95 or upper 48 bits.
- Double-Precision - Uses output bits 0 through 47 or lower 48 bits.
- Single-Precision with Coefficient Overflow - Performs a wired right shift one of the upper bits and selects bits 49 through 96.
- Double-Precision with Coefficient Overflow - Performs a wired right shift one of the lower bits and selects bits 1 through 48.

No matter which of these four paths are accepted they become bits 0 through 47 of the Xi Register in the CPU.

Let's look at four of the five signals that control the functions that are on the diagram FAD 3.0 at coordinate C1/C2. In quadrant B is a FM module at 5C06. In the lower right portion of the Output Selection Control logic block are the three signals needed: SAMPLE UPPER HALF, SAMPLE CENTER, and SAMPLE LOWER HALF. In the actual translation of each signal you have a common term, $\overline{\text{GO ADD SPECIAL CASE}}$. This term as shown in the logic block can happen only if the operand has Overflow, Indefinite, or Underflow, which is caused by subtracting 60 from one exponent during a double precision operation. SAMPLE UPPER HALF is used if there is no special case, GO and it is not double precision. SAMPLE CENTER is used if there is no special case and GO. SAMPLE CENTER is always used either in single- or double-precision operation. SAMPLE LOWER HALF is used if there is no special case, GO and there is double-precision. The fourth signal is second from the top. It is DP or DOUBLE PRECISION. All four signals go to the FK modules on the FAD 3.2 diagrams at coordinate E1/E2 in quadrant D.

On the FK module at 5E06 is the remaining signal needed, Coefficient Overflow at the bottom. It is generated by the FL module at 5D12 in quadrant A.

Let's use this FK module at 5E06 as the example as it is the only one that has all four possible outputs. This module handles bits 45 through 53 of the 97-bit result. Using the example that was stated at the start, there is single precision and no Overflow. Looking at the three commands in the lower left of the FK module at 5E06, they are: $\overline{\text{DP}}$, SAMPLE CENTER, and $\overline{\text{COEFFICIENT OVERFLOW}}$. This would cause DP on the diagrams to be equal to a one with SAMPLE CENTER being a one. This one would go to the top AND gate. With no Coefficient Overflow, $\overline{\text{COEFFICIENT OVERFLOW}}$ is complemented to a one also sent to the top AND gate. Complemented, it goes to the top AND NOT function. This complements the data or bits 48 through 53 from the adder to the Xi Register in the CPU. Since it was single precision, SAMPLE UPPER HALF would be set gating bits 54 through 96 to the Xi Register. Read note 4 . Following are the four possible combinations. Make sure that you understand and can complete all the necessary gates.

- IF $\overline{\text{DP}}$ and SAMPLE CENTER are equal to ones and Coefficient Overflow is equal to a zero, the data path for bits 48 through 53 is ENABLED (this was the example just covered).

- If DP, and SAMPLE CENTER are equal to ones and Coefficient Overflow is equal to a zero, the data path for bits 45 through 47 is ENABLED.

- If $\overline{DP}$, SAMPLE CENTER and Coefficient Overflow are equal to ones, the data path for bits 49 through 53 Shifted is ENABLED.
- If DP, SAMPLE CENTER, and Coefficient Overflow are equal to ones, the data path for bits 45 through 48 Shifted is ENABLED.

This completes the coefficient for the four clock periods and a normal operation.

Third Clock Period: Exponent. Take the FAD 3.0 diagram at coordinate C1/C2, quadrant B and the FE modules at 5C05. At the end of the second clock period, the Xk exponent equal to 3723 was being held in the selected Exponent Registers on the FE module at 5C05 and FD module at 5C04. The output of these two registers goes to an adder on a FM module at 5C06. This adder is called Exponent minus 60_8 or zero. If doing a double-precision you will make the gate at the bottom or the adder, allowing a 60_8 to be subtracted from the exponent. In either case the result is held in the Exponent Holding Register and is back to true form. With no subtraction this value is equal to 3723.

Fourth Clock Period: Exponent. The output of the Exponent Holding Register is sent to an FN module at 5C07 on FAD 3.2 diagrams at coordinate E1/E2 in quadrant B. The exponent can go two different ways. These paths are controlled by SAMPLE EXPONENT. The signs of the two operands come from the FM module at 5C06 on FAD 3.0 diagrams and BORROW TO 96 comes from the FL module at 5D12 in quadrant A. SAMPLE EXPONENT is generated by GO ADD SPECIAL CASE·GO. SAMPLE EXPONENT controls both output paths so if it is equal to zero both paths will be blocked. Since this is an AND NOT function all ones are output to the Xi Register.

BORROW TO 96 with the signs of the two operands tell the FN module two things. As the examples and exercises show, if the two signs are alike the BORROW TO 96 determines if there is Coefficient Overflow. If Coefficient Overflow occurs the exponent has one added to it because the coefficient was right shifted by one. Path Output No. 2, the top half of the FN module, is used if there is no Coefficient Overflow. If there is Coefficient Overflow, path Output No. 1 in the bottom half of the FN module is used.

The following combinations of signs and BORROWS TO 96 cause the exponent to take the path through Output No. 2.

$\left.\begin{matrix}0\\0\end{matrix}\right\}$ Positive	$\left.\begin{matrix}1\\1\end{matrix}\right\}$ Negative	$\left.\begin{matrix}1\\0\end{matrix}\right\}$ Unlike
No BORROW TO 96 or BORROW TO 96	A BORROW TO 96 or $\overline{\text{BORROW TO 96}}$	A BORROW TO 96 or $\overline{\text{BORROW TO 96}}$
The sign of the result is positive. No Coefficient Overflow exists.	The sign of the result is positive. No Coefficient Overflow exists.	The sign of the result is positive. No Coefficient Overflow exists.

When both signs are positive and BORROW TO 96 is equal to one (which means no BORROW to bit 96), no Coefficient Overflow exists, and the result is positive. Complement Control No. 2 adds the bias by complementing bits 0 through 9. Sign No. 2 gives the proper sign to the coefficient bit 59. Output Control No. 2 gates the exponent to the Xi Register.

When both signs are negative and BORROW TO 96 is equal to zero (which means a Borrow to Bit 96), no Coefficient Overflow exists and the result is negative.

Complement Control No. 2 adds bias by complementing bit 10. Sign No. 2 and Output Control add the sign and gate as explained for both positive signs.

When the signs are not alike and BORROW TO 96 is equal to zero (which means to Borrow to bit 96), no COEFFICIENT OVERFLOW exists and the result is positive. Bit 10 is complemented to add bias. The sign is added to the coefficient and the output is gated to Xi.

Let's look at two examples: both positive and both negative.

In the first example both are positive and there is no BORROW TO 96. The example that has been used from the start can be used here.

Xj = 1717 5431 1111 1111 1111

Xk = 1723 4311 1111 1111 1111

Looking at this example in learning activity 8-A these two numbers both have positive coefficients. At the beginning of this operation the exponent was stored in the Exponent Holding Register as 3723 on the FM module at 5C06. There is no BORROW TO 96 so BORROW TO 96 equals 1, meaning no Coefficient Overflow. This 3723 enters the FN module at

5C07. Since both signs are positive and BORROW TO 96, Output Control No. 2 is used. With both signs positive Complement Control No. 2 will be a zero causing bits 0 through 9 to be complemented giving you 2054. Sign No. 2 will be a one. This gives you a value of 6054 just to the left of the AND NOT function. When Output Control No. 2 is a one the output is gated and complemented sending a value of 1723 to the Xi Register.

In the second example both are negative and there is a BORROW TO 96 (actually, a $\overline{\text{BORROW TO 96}}$).

Using an example of Xj = 6002 2000 0000 0000 0000 and Xk = 6003 4000 0000 0000 0000, both coefficient signs are negative:

```
    Xj = 6002 is complemented to 1775 so  1777
                                         -1775
                                         -----
    Xj Exponent equals minus                 2

    Xk = 6003 is complemented to 1774 so  1777
                                         -1774
                                         -----
    Xk Exponent equals minus                 3
```

This makes Xj larger and Xk will be RS one.

```
    Xj =      111 ¦ 7000  0000  0000  0000∧7777  7777  7777  7777
    Xk RS 1   111 ¦ 6000  0000  0000  0000∧3777  7777  7777  7777
              ----------------------------------------------------
            ┌ 111 ¦ 5000  0000  0000  0001∧3777  7777  7777  7776
            └►EAB ¦                                             1
              ----------------------------------------------------
              111 ¦ 5000  0000  0000  0001∧3777  7777  7777  7777
```

Xj being the largest, 6002 is stored in the Xi Register.

Xj's exponent would have been stored as 3775 in the Exponent Holding Register on the FM module at 5C06. Since either sign is negative and have $\overline{\text{BORROW TO 96}}$ Output Control No. 2 is used again. The value of 3775 enters the module. Complement Control No. 2 equals a 1 as both signs are negative, so only bit 10 is complemented, giving a value of 1775. Sign No. 2 equals a 0 so the value to the left of the AND NOT will equal 1775. When Output Control No. 2 is a 1, the output is gated and complemented sending a value of 6002 to the Xi Register.

The following combinations of signs and BORROWS TO 96 cause the exponent to take the path through Output No. 1 and the Exponent +1/0 Adder.

$\left.\begin{matrix}0\\0\end{matrix}\right\}$ Positive	$\left.\begin{matrix}1\\1\end{matrix}\right\}$ Negative	$\left.\begin{matrix}1\\0\end{matrix}\right\}$ Unlike
A BORROW TO 96 or $\overline{\text{BORROW TO 96}}$. The sign of the result is positive. Coefficient Overflow exists.	No BORROW TO 96 or BORROW TO 96. The sign of the result is negative. Coefficient Overflow exists.	No BORROW TO 96 or BORROW TO 96. The sign of the result is negative. No Coefficient Overflow exists.

When both signs are positive and BORROW TO 96 is equal to a zero (meaning a Borrow to bit 96), there is Coefficient Overflow. The exponent has one added to it. Complement Control No. 1 again adds bias by complementing bits 0 through 9. It ensures that the exponent correction for Coefficient Overflow does not result in the logic from sending a minus 0 exponent. For example, if the exponent was equal to -1 the addition of 1 would result in a value of negative zero. Complement Control No. 1 senses this and will output a plus zero. The result is positive. Complement Control No. 1 adds the bias by complementing bits 0 through 9. Sign No. 1 gives the proper sign to the coefficient bit 59. Output Control No. 1 gates the exponent to the Xi Register. Plus one goes to the adder any time the signs are alike.

When both signs are negative and BORROW TO 96 is equal to a one (meaning no BORROW TO 96), there is coefficient overflow and the exponent has one added to it. Complement Control No. 1 adds bias by complementing bit 10 in this case. The remaining controls are as previously explained.

When the signs are not alike and BORROW TO 96 is equal to one (meaning no BORROW TO 96), no Coefficient Overflow exists and the result is negative. Complement Control No. 1 adds bias by complementing Bit 10. There is no plus one to the adder. The sign bit is added and result is sent to the Xi Register.

Let's look at these three cases.

First, both are positive and there is a BORROW TO 96 ($\overline{\text{BORROW TO 96}}$). Using the second example in learning activity 8-A you have the condition needed. Xj's exponent equals 2000 or +0 and Xk's exponent equals 1776 or -1. As previously seen with position coefficients, bit 10 is complemented to unbias the exponent, so 0000 is stored in the Exponent Holding Register. This 0000 enters the FN module and because both signs are positive and $\overline{\text{BORROW TO 96}}$, Output Control No. 1 is used. With both signs alike it causes +1 to the adder giving you a value of 0001.

With both signs positive and the largest exponent not equal to -1, Complement Control No. 1 equals zero, causing bits 0 through 9 to be complemented giving you 1776. Sign No. 1 will equal one. This becomes bit 11 of the exponent making it a value of 5776, just to the left of the AND NOT function. When Output Control No. 1 is a one this value is gated and complemented sending a value of 2001 to the Xi Register.

In the second example both signs are negative and there is no BORROW TO 96 (BORROW TO 96).

Using an example of Xj = 6002 1000 0000 0000 0000 and
Xk = 6003 0400 0000 0000 0000, both coefficient signs are negative:

```
Xj = 6002 is complemented to 1775 so  1777
                                     -1775
Xj's Exponent equals minus               2

Xk = 6003 is complemented to 1774 so  1777
                                     -1774
Xk's Exponent equals minus               3
```

This makes Xj larger and Xk will be RS one

```
Xj        =  111 ¦ 1000  0000  0000  0000 ∧ 7777  7777  7777  7777
Xk RS 1   =  111 ¦ 4200  0000  0000  0000 ∧ 3777  7777  7777  7777
             110 ¦ 5200  0000  0000  0001 ∧ 3777  7777  7777  7776
             EAB ¦                                               1
             110 ¦ 5200  0000  0000  0001 ∧ 3777  7777  7777  7777
```

Xj being the largest, plus one or 6001 is stored in the Xi Register.

Xj's exponent is stored as 3775 in the Exponent Holding Register. This 3775 enters the FN module and because both signs are negative and BORROW TO 96, Output Control No. 1 is used. Both signs alike causes a +1 to the adder giving a new value of 3776. With both signs negative and the largest exponent not equal to -1 Complement Control No. 1 equals one. This will cause bit 10 to be complemented, giving you a value of 1776. Sign No. 1 will equal zero. This becomes bit 11 of the exponent so nothing has changed. The output goes through the AND NOT function causing the value to be complemented to a value of 6001 and this is stored in the Xi Register.

In the third condition signs are unlike and no BORROW TO 96 (BORROW TO 96). Using exercise question No. 2 in learning activity 8-A you have the conditions needed. Xj's exponent equals 2000 or +0 and Xk's exponent equals 5771 or +6. Xj has a positive coefficient and Xk's coefficient is negative. Xk is the largest and with a negative sign its exponent is unbiased by complementing bits 0 through 9. The value of 0006 is stored in the Exponent Holding Register. This value is sent to the FN module and because one sign is negative and BORROW TO 96, Output Control No. 1 is used. Since signs are not alike there is no +1 so the output of the adder equals 0006. With signs not equal Complement Control No. 1 equals one, causing bit 10 to be complemented. It now has a value of 2006. The sign bit will be equal to zero because one sign is negative. Just to the left of the AND NOT function is a value of 2006.

When gated through, it is complemented so the value of 5771 is stored in the Xi Register. This completes a normal 30 instruction. Still to be covered are Double-Precision Sum, Rounding Sum, and Special Case. Review the past material if any part of Floating Add is not clear.

If all is clear, let's look at what is needed to execute a 32 instruction or Floating Double Precision Sum. The control of the coefficient during double precision was covered during normal operation.

Fourth Clock Period: Coefficient on page 8-43. This leaves exponent correction. From the exercises completed during learning activity 8-A you learned that 60_8 is subtracted from the largest exponent because you are using the lower 48_{10} bits of the resulting coefficient. Coefficient Overflow still exists so there could be an exponent correction of +1. Select the FAD 3.0 diagram at coordinate C1/C2 in quadrant D. There is an FD module at 5C04. The fifth input from the bottom is ml and if equal to a one it means a 32 instruction. This bit is received from the CIW Register during the first clock period and is stored in a Clock Holding F/F. During the third clock period exponent bits 0 through 10 enter the Exp -60/0 adder on the FM module at 5C06 in quadrant B. This value is complemented so it can be subtracted. During this time the signal DP will equal a one, gating in the value of 60_8. For example, if the exponent equals 3723, this would be complemented to equal 0054. The 60_8 results in a value of 0134. Since the output of this adder is always in true form the value would be 3643 or 60_8 less than its input, and be stored in the Exponent Holding Register.

Let's look at a 34 instruction or Round Floating Sum. During this operation you set bit 2^{-1} or bit 47 of the 96-bit coefficient to the complement of the sign bits. The round bit is always added according to the following rules.

- It is inserted in coefficient with largest exponent.

- It is inserted in Xk's coefficient if both exponents are equal.
- A second round bit is added to the other coefficient if both operands are normalized or have unlike signs.

In quadrant A on the FF module at 5B07 the bottom input is m2. If it is equal to a one, it means a 34 instruction and m2 Input F/F would set on the first clock period. At this time let's just add a round bit as the first two rules dictate. This bit is now Round Reference and goes to the FC modules at 5C02 or 5C03. If the sign bit is negative and equal to a one, then the round bit is complemented.

The FO module at 5B09 on FAD 3.1 diagram at coordinate D1/D2 quadrant C has a Reference Round Bit opposite the coefficient sign for both Xj and Xk. The bottom two inputs of the FO module are Reference Xk and Xj Round Bit. These two inputs are gated by SIGN OF DIFFERENCE, which equals one if Xj is the largest exponent. So if Xk is largest or equal to Xj, the SIGN OF DIFFERENCE would equal zero. This would gate Reference Xk Round Bit along with Xk bits 0 through 5 to the Reference Operand Register. These become bits 47 through 53 of the 96-bit coefficient. If Xj was the largest then Reference Xj Round Bit and Xj bits 0 through 5 would be gated to the Reference Operand Register.

The third rule involves adding a round bit to the second operand. Return to the FF module at 5B07 in coordinates C1/C2 and quadrant A. Just above the m2 Input F/F is the logic that tests for both operands normalized or having unlike signs. For normalization of each operand the logic checks bit 59 and bit 47 and also checks the sign bits for not being equal. If either is the case, the Gate Round Shift and Round Reference is made. These go to the FC modules at 5C02 and 5C03. Both bits would now be the complement of the sign bit. These bits are sent to the FAD 3.1 diagrams. The shifted round bit goes to FG module at 5B08 or FO module at 5B09. Reference Round Bit goes to the FO module at 5B09. This gives a round bit to both operands.

The remaining item to consider in Floating Add is Special Case.

The Floating Add Unit senses three special cases: Overflow, Underflow and Indefinite. Using the FAD 3.0 diagrams at coordinates C1/C2 and quadrant A there are two FC modules at 5C02 and 5C03. Using 5C02 as an example, it checks for Xk Overflow and Xk Indefinite. These results are sent to the FD module at 5C04 in quadrant D, and are stored for use during the third clock period. This module has a Flip Flop for both Xk and Xj Overflow but the Xj and Xk Indefinite are ORed to the second clock period. In quadrant B is a FM module at 5C06. It has the logic for Special Case testing. This logic tests for these conditions during the third clock period and sends the correct special case flag to X Sign Control Translator in the CPU.

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-C

Let's review the three special case conditions.

OVERFLOW

This condition involves an exponent value of positive 1777 (3777 or 4000 in packed form). This is the largest exponent value that can be expressed. This value may result from a calculation. If so this situation is called a Partial Overflow. Any further computation using this result causes Overflow.

A complete Overflow occurs when a result requires an exponent value larger than positive 1777.

UNDERFLOW

This condition involves an exponent value of negative 1777 (0000 or 7777 in packed form). This is the smallest exponent value that can be expressed. This value may result from a calculation. If so, this situation is called Partial Underflow. Any further computation using this result causes Overflow.

A complete Underflow occurs when a result requires an exponent value smaller than negative 1777.

INDEFINITE

This condition involves an exponent from which a calculation cannot be made. It has a value of negative 0000 (1777 or 6000 in packed form).

NONSTANDARD OPERANDS

In summary, the special operands in packed format are:

●	Positive Overflow	(+ ∞)	3777 X X
●	Negative Overflow	(- ∞)	4000 X X
●	Positive Indefinite	(+ IND)	1777 X X
●	Negative Indefinite	(- IND)	6000 X X
●	Positive Underflow	(+ 0)	0000 X X
●	Negative Underflow	(- 0)	7777 X X

Figures 8-8 and 8-9 show the resulting forms when various combinations of Overflow and Indefinite forms are used. The designation W is any word except ± ∞ and ± IND.

		Xk			
		W	+ ∞	- ∞	+ IND
Xj	W		+ ∞	- ∞	IND
	+ ∞	+ ∞	+ ∞	IND	IND
	- ∞	- ∞	IND	- ∞	IND
	± IND	IND	IND	IND	IND

Figure 8-8. Xj Plus Xk (30, 32, 34 Instructions)

		Xk			
		W	+ ∞	- ∞	+ IND
Xj	W		- ∞	+ ∞	IND
	+ ∞	+ ∞	IND	+ ∞	IND
	- ∞	- ∞	- ∞	IND	IND
	± IND	IND	IND	IND	IND

Figure 8-9. Xj Minus Xk (31, 33, 35 Instructions)

Figure 8-10 shows the flag sent to X Sign Control Translator when various combinations of Overflow and Indefinite forms are used. For example, if either operand is Indefinite, the Indefinite Flag is set equal to one. If the Xk and Xj operands are negative overflows (-), the Overflow Flag is set equal to one.

		Xk			
		W	+ ∞	- ∞	IND
Xj	W		OVF	OVF	IND
	+ ∞	OVF	OVF	IND	IND
	- ∞	OVF	IND	OVF	IND
	IND	IND	IND	IND	IND

Figure 8-10. Signs Are Reference and Shifted Signs

An Underflow Flag is generated when the unbiased exponent is a value more negative than -1777 and the instruction is Double-Precision. This happens as 60_8 is subtracted from the exponent. If this happens, the FM module sends an Underflow Flag to X Sign Control Translation. The logic mainly looks at bit 10 of the exponent and DP to generate Underflow.

An Overflow Flag is generated when the unbiased exponent of either Xj or Xk is a value of plus 1777 and no Indefinite.

An Indefinite Flag is generated when the unbiased exponent of both Xj and Xk are equal to a value of plus 1777, but the signs of their coefficients are not equal, or either Xk or Xj has an unbiased exponent with a value of -0000. This completes all the logic for Floating Add.

You have completed learning activity 8-C. You may review this material or continue with the next learning activity.

LEARNING ACTIVITY 8-D. PROGRAMMED TEXT: FLOATING ADD TROUBLESHOOTING

This activity enables you to develop approaches to troubleshooting the Floating Add Unit.

OBJECTIVE

- You will be able to analyze hypothetical failures and come to a logical solution.

Directions: Place microfiche GF60420300W Number 14 (4 of 10) in your microfiche viewer and select coordinate C1/C2.

This exercise gives you practice in troubleshooting problems in the Floating Add Functional Unit.

Directions: Each problem has the instruction that was executed. It also gives you its source operand(s) and the result(s).

List as many malfunction(s) as possible that could cause the following results. List the.module(s) location and, if possible, the gate or control name.

Check your answers with those given at the end of the learning activity.

1. Instruction execution was a 30123.

```
Given:   X2 = 4000 7777 7777 7777 7777
         X3 = 4000 7777 7777 7777 7777
         X1 = 1777 0000 0000 0000 0000
```

5 C06 2 F AN7

X1 = 4000 7777 7777 7777 7777

NEG. OVERFLOW

CYBER 170 Model 750/760 Functional Units
Learning Activity 8-D

2. Instruction executed was a 30123.

```
Given:  J X2 = 1720 4444 4444 4444 4444
        K X3 = 1723 5000 0000 0000 0000
        I X1 = 1723 5000 0000 0000 0000
```

```
 1777      1777
 1720      1723
-  57     -  54
```

04 444 4444 4444
30 000 0000 0000

1723 5 444 4444444 4444

X_2

SC05

3. Instruction executed was a 30123.

```
Given:  J X2 = 5776 6000 0000 0000 0000
        K X3 = 5770 3000 0000 0000 0000
        I X1 = 6051 0077 7777 7777 7777
```

5776   5770
2001   2007
2001

SC04
SC06

111 6000 0
111 773 000|00 7
111 573 0000777 0
57 60100 6

2007
\- 0
27

4. Instruction executed was a 31345.

```
Given:  K X5 = 6053 1777 7777 7777 7777
        J X4 = 1724 5000 0000 0000 0000
        I X3 = 1724·3000 0000 0000 0000
```

K 1724   1724

0/6000000 0

5000000 0

3000000 0

1724
1
1725   54   54000—   5D12

5. Instruction executed was a 32345.

```
Given:  K X5 = 0040 3777 7777 7777 7777
        J X4 = 0060 0000 0000 0000 0000
        I X3 = 0000 0000 0000 0000 0000
```

1777   1777
0040   0060
-1737   -1710
60

5C04
5C06
5C05

37

00000   1777   17777#

0000 777774 0000

6. Instruction executed was a 33456.

```
Given: K X6 = 5734 7777 7777 7777 7776
       J X5 = 2040 0000 0000 0000 0000
       I X4 = 1762 7000 0000 0000 0777
```

2043 2040

~ 0000 0000 0000 0000 0001

2043
7740
1762

5F01

ANSWERS FOR LEARNING ACTIVITY 8-D

In the answers to these problems we have included test points, pin numbers, or terms. It is not necessary to know them, but for students who wish to increase their knowledge of troubleshooting, we have included them.

1. The correct answer is:

   X1 = 4000 7777 7777 7777 7777

   Possible malfunction(s):

   a. 5C06 - Special Case Detection Network FM Module TP62 or 66 = 1

2. The correct answer is:

   X1 = 1723 5444 4444 4444 4444

   Possible malfunction(s):

   a. 5C05 - Shift 128 Gate on FE Module, can be checked on any FH module TP 71, 72, 73, and 74 All = 1

3. The correct answer is:

   X1 = 5770 2760 0000 0000 0001

   Possible malfunction(s):

   a. 5C04 - Select Double Precision result FD module TP66 or TP 64 = 0

   b. 5C06 - Select Double Precision result FM module TP74 = 0

4. The correct answer is:

   X1 = 1725 5400 0000 0000 0000

   Possible malfunction(s):

   a. 5D12 - Coefficient Overflow Detector on FL module

5. The correct answer is:

   X3 = 0000 7777 7400 0000 0000

   Possible malfunction(s):

   a. 5C06 - Detect Underflow on FM module TP15 or TP16 = 0

   b. 5C05 - Shift 128 gate on FE module - can be checked on any FH module TP71, 72, 73, and 74 all = 1

   c. 5C04 - Shift 16 signal lost on FD module - TP41 and TP46 = 1

6. The correct answer is:

   X4 = 1762 7000 0000 0000 0000

   Possible malfunction(s):

   a. 5E01 No Transmit Gate on FK module -
      TP01, 03, 04, or 06 = 1

INSTRUCTOR CONFIRMATION

INSTRUCTIONS: When all objectives in the preceding laboratory session have been met, complete this form. Please fill in all the information asked for in each blank space. Next, the form must be signed by the lab instructor in the space provided. Then, remove this page and fold it so that the address on the back appears. Then staple it and mail it. When this form is received by ESE, the student's PLATO records will be updated to reflect mastery of this lab module.

STUDENT NAME: ______________________________
(print)

EMPLOYEE NUMBER: ___________________________

DEPARTMENT NUMBER: _________________________

PLATO SIGN-ON NAME: ________________________
(print)

PLATO GROUP NAME: __________________________

I certify that the above named student has satisfactorily completed all procedures and has met the objectives of __________________ in the __________________________ course.

INSTRUCTOR NAME: ______________________________
(print)

INSTRUCTOR SIGNATURE________________________DATE:____________

COURSE TITLE: CYBER 175 FUNCTIONAL UNITS
PRODUCT NUMBER: R0502

INSTRUCTOR COMMENTS:

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Engineering Services Education
1009 Washington Avenue North
Minneapolis, Minnesota 55401

ATTN: Administration of Individualized
Instruction (NTHESE)

CUT ALONG LINE

FOLD FOLD

COMMENT SHEET

MANUAL TITLE CYBER 175 FUNCTIONAL UNITS VOLUME 1

PUBLICATION NO. 75445737 REVISION A Prod. No. R0502

FROM: NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual.

CUT ALONG LINE

PRINTED IN U.S.A

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241
MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Engineering Services Education
P.O. Box 0
Minneapolis, MN 55440

ATTN: EDUCATION QA
NTHESE

CUT ALONG LINE

FOLD FOLD